A Streaming Piecewise-Constant Model

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

Abstract

The piecewise-constant image model (PWC) is remarkably effective for compressing palette images. This paper discloses a new streaming version of PWC that retains the excellent compression efficiency of the original algorithm while dramatically enhancing compression performance. Further, compression throughput is made more constant, making it possible to code sparse images very quickly.

1. Introduction

The piecewise-constant image model[1] is a remarkably effective method for compression of palette images. PWC outperforms standard techniques on all types of palette images, and is particularly effective on images that exhibit significant two-dimensional structure. This year's DCC call for papers[*] contains 17 images of the type that has seen widespread use on the Internet. PWC compresses these image exceptionally well and it was impossible to resist using them in an introductory comparison. Table 1 shows the total byte count for all 17 images when uncompressed, and when compressed with GIF, PNG, and PWC.

BMP	GIF	PNG	PWC
251,986	25,542	19,733	8,275

Table 1
Compression of Images from the 1999 DCC Call for Papers

Although initially targeted as an Internet graphics delivery medium, PWC suffers from two problems that limit its potential in interactive applications. The most important limitation in original PWC (hereafter designated PWC2) is that is makes two separate image passes. One pass establishes rectilinear domain connectivity and a second pass establishes domain color.

The second drawback to PWC2 is that it is a completely unary method and must code at least one binary decision for every image pixel. This puts a floor on compression time that is proportional to image size, not image complexity. This is unacceptable in that many palette images are quite sparse and code extremely fast with conventional methods.

This paper describes a new streaming version of PWC. Streaming is affected by making two passes over each scanline instead of over the entire image. Compression speed is enhanced by a new method for skipping over uniform portions of an image. The skip mechanism simultaneously improves compression efficiency and eliminates high-skew decision streams. The mechanism has general utility.

[*] http://www.cs.brandeis.edu/~dcc/

2. Two-Pass PWC

The piecewise-constant image model composes an image from rectilinearly connected domains each having a constant color. Boundaries between domains and domain colors are described with binary decisions. The four types of decisions made when describing a PWC model are shown in Table 2.

Q1	Is the current pixel’s color identical to that of a specified rectilinearly connected neighbor?
Q2	Is the current pixel’s color identical to that of a specified diagonally–connected neighbor?
Q3	Is the current pixel’s color identical to a guessed value?
Q4	What is the current pixel’s color?

Table 2
Piecewise–Constant Language

In the PWC2 coder, domain boundaries are established in a first image pass and domain colors are established in a second pass. With the two pass technique, the color of a domain need only be acquired once. For example, Figure 1 shows a fragment of the JPEG-LS compound document #1. Even though the fragment contains only two colors, it is embedded in a 256 color image precluding simple color assumptions.

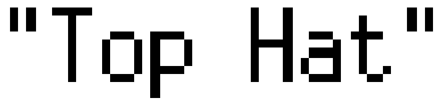

Figure 1
Excerpt from JPEG-LS
Compound Document #1

In the two-pass technique, the extent of concave domains such as the ‘H” of Figure 1 is completely defined in the first pass. When such a domain is initially encountered in the second pass, its color is established with decisions **Q2-Q4** and then completely propagated throughout the domain via a flooding process.

3. Streaming PWC

The basic strategy of streaming PWC is to make two passes through each scanline. The first pass makes **Q1** decisions to establish rectilinear connectivity within the scanline and to the previous scanline. The second pass determines the color of each domain *stripe* by making **Q2-Q4** decisions.

The requirements for the color determination pass are somewhat subtle. The key point is to avoid making unnecessary color decisions. For example, once the color for the roof of the “**T**” in Figure 1 is established on the first scanline it is possible to propagate the color down the trunk without making further color decisions. Similarly, though the color of the “**H**” must be determined twice on the first scanline, color propagation is possible on subsequent scanlines. Perhaps not as obvious is that color propagation is also possible for the cross of the “**t**”.

Streaming PWC maximizes color propagation by making two color determination passes over each domain stripe. The first pass does not make any coded decisions. On the second pass, the boundary model built by the connectivity pass is consulted to determine whether or not a color can be propagated from the previous scanline. If a pixel on the previous scanline is not separated from the current scanline by the boundary model, its color becomes the propagation color. Colors lying above horizontal edges in the boundary model are marked as impossible.

If color propagation is not possible, impossible color information is then used to filter **Q2** and **Q3** decisions in a **Q2-Q4** color determination sequence. The streaming algorithm is summarized thusly:

- For each image scanline
  - Make a first pass and make **Q1** decisions to determine rectilinear connectivity.
  - On a second color determination pass, for each domain stripe:
    - Make a first pass to determine the possibility of color propagation and mark impossible colors.
    - If color propagation is not possible, determine the stripe color via a **Q2-Q4** sequence that skips impossible **Q2** and **Q3** decisions.
    - Make a second pass along the stripe to update the color model.

4. Performance Considerations

Several performance enhancements make PWC significantly faster than PWC2. The enhancements focus on reducing the number of coded decisions. A secondary consideration is reducing the necessary precision in the statistical coder.

4.1 Connectivity Constraints

For conditioning **Q1** decisions, both PWC2 and PWC use the edge model proposed by Tate[2]. At each separator lattice location, **L**, a **Q1** decision is made to determine the presence or absence of a vertical edge. Once the vertical edge state is known, connectivity constraints often deterministically determine the horizontal edge state. To the extent that determinism operates, the number of decisions that must be statistically coded is reduced.

The following analysis develops an upper bound on the rectilinear decision entropy. If p is the zero order probability that a separator lattice site is full, then the probability that a horizontal separator is deterministically determined is:

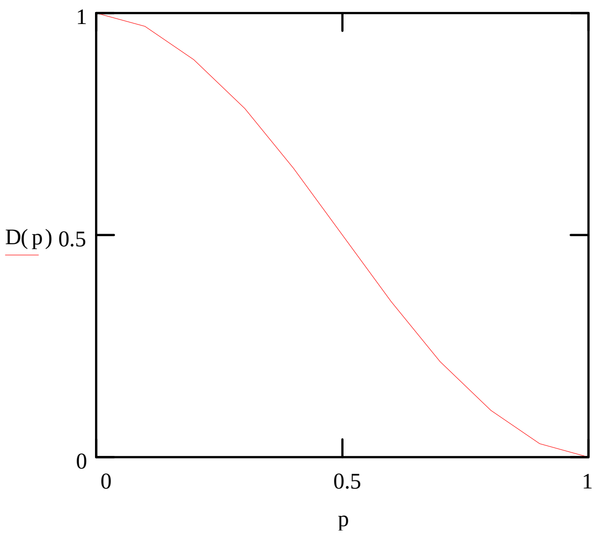

Figure 2
Deterministic Decision Probability

$$D(p) = (1-p)^3 + 3p(1-p)^2 \,. \qquad (1)$$

$D(p)$ is plotted in Figure 2.

Given p as before, the vertical decision entropy is:

$$H_1(p) = -p\log_2(p) - (1-p)\log_2(1-p)\,. \tag{2}$$

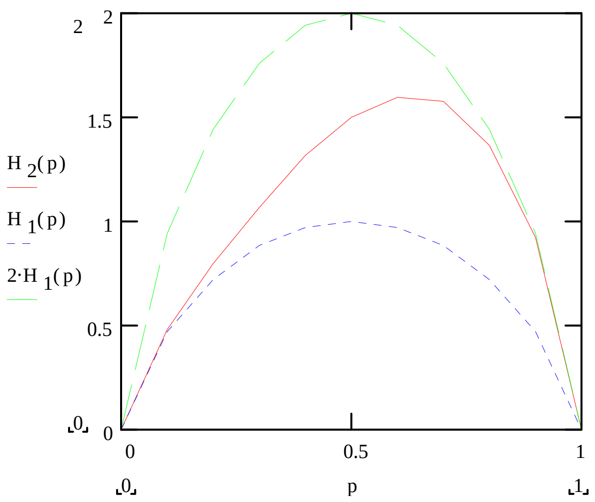

Figure 3
Total Edge Map Decision Entropy

The horizontal entropy adjusted for determinism is:

$$H_h(p) = (1-D(p))H_1(p)\,, \tag{3}$$

and the total edge entropy is

$$H_2(p) = H_1(p) + H_h(p)\,. \tag{4}$$

The maximum of H_2 is 1.607 and it occurs at a full edge probability of 0.632. H_2 is plotted in Figure 3. Note how H_2 approaches the single decision entropy, H_1, at low p and approaches $2 \cdot H_1$ at high p. On sparse images only one **Q1** decision need typically be made at each separator lattice location.

4.2 Skipping Inactive Areas

One decision is better than two, but PWC2 must code at least one decision for every image pixel even if the image is completely uniform. Since palette images often contain large uniform areas that code very fast with conventional methods, streaming PWC contains a new hybrid unary/run length model that accelerates coding of sparse images.

The surprisingly simple solution introduces a new decision to the PWC language. An *inactive* location is defined as a location where the edge decision context model contains no edges. During an edge scan when an inactive location is encountered, PWC makes a decision that determines whether or not to skip to the next *active* location. The new decision, designated **Q0**, is shown in Table 3. **Q0** decisions are made under a special skip context.

Q0	Skip to the next active location?

Table 3
New Skip Decision

If a **Q0** decision is made to the affirmative, the algorithm skips forward to the next active context and continues normal processing. If a skip occurs, the edge model need not be updated since the skipped locations remain inactive. If a skip cannot be made, PWC temporarily disables skip processing until the next edge is discovered through normal unary processing.

The skip mechanism actually yields better compression efficiency than the completely unary model. The reasons for this are two fold. First, skip decisions are relatively infrequent. And second, inactive edge decisions are now made in two contexts, the skip context and the normal inactive context.

4.3 A Better Predictor for Q4

PWC2 uses a predictive model with multiple activity classes[3] for **Q4**. The three closest causal pixels are used to select the activity class. The predictor used is the single previous pixel on the same scanline. While this simple predictor works reasonably well for typical palette image, it is unsatisfactory for grayscale images.

Many predictors that work well for grayscale images misbehave on typical palette images. One predictor that performs well on both types of images is that used in LOCO-I[4]. The LOCO-I predictor consists of a three pixel planar predictor augmented with a simple edge detector.

Using this predictor, streaming PWC compresses grayscale images significantly better than PWC2. Compression of typical palette images is also improved by approximately 0.5%. Better prediction also leads to slightly fewer statistically coded decisions and somewhat improved performance.

4.4 Low Precision Arithmetic Coding

Both versions of PWC use Don Speck's carry-free coder[5] for statistical coding of binary decisions. This coder goes back to the roots of arithmetic coding to avoid IP issues associated with more modern coders. Statistics are kept as counts and the coding interval is split with a multiply/divide operation. Both versions of PWC augment the basic coder with 3-4-5 adaptation where statistics are halved whenever the lps count reaches six.

The maximum count value used in PWC2 is 65,535. This is to accommodate the high skew of the inactive edge decision context. With its skip model, streaming PWC can get by with a maximum skew of 255. This allows for a $16 \times 8 \div 8 = 16$ bit multiply/divide operation and reduces the necessary Intel Pentium CPU cycles from 52 to 36.

The combination of the skip model with accurate coding of low skews also produces excellent compression. In fact, as is shown in the experiments, the combination does better than JBIG on the CCITT test documents. This is probably because state driven arithmetic coders such as the QM code of JBIG exhibit small inefficiencies as decision probability approaches 50%. Several recent efforts have focused on this problem[6],[7].

5. Depth One Model Optimization

The edge model is overkill for depth one images, and to attain reasonable speed streaming PWC uses a 10 pixel neighborhood color model on two color images[8]. In the reduced model, the inactive criteria becomes either uniformly white or uniformly black. Statistics for the two "colors" of skip decisions are kept separately. On the CCITT test documents, a white only inactive criteria yields better compression than the symmetric criteria. This is because uniform black areas are relatively short.

In this environment, the skip information is mostly wasted. For example, if a uniform black area is only one column wide, a negative skip decision determines that the next pixel is white. The wasted information can be recaptured by making non-skipped color decisions in more than one context. The base two logarithm of the remaining skip length works well as a conditioning function.

6. Experiments

The major new consideration for streaming PWC is that the color of a concave domain may have to be acquired multiple times. Intuition indicates that reacquisition should occur relatively infrequently in practice. To verify this assumption, it is instructive to compare streaming PWC against the original two-pass algorithm (PWC2) on the palette image corpus. The results of Table 4 show the average compression achieved by streaming PWC to be only 0.6% worse than a slightly improved version of PWC2.

The version of PWC2 used in this test does not include all the improvements incorporated into streaming PWC. In particular, streaming PWC contains the skip mechanism, an improved predictor, and improved tracking of impossible colors. Nevertheless, the actual coding penalty for color reacquisition is small. The author's estimate is somewhere between two and three percent for the palette image corpus.

Image	PWC2	PWC
benjerry	2,387	2,473
books	8,630	8,719
ccitt01	12,890	12,685
cmpndn	40,390	40,354
cmpndu	53,951	53,725
flax	149	171
gate	15,530	15,671
music	735	721
netscape	10,649	10,839
pattern	1,174	1,096
sea_dusk	657	696
stone	4,001	3,982
sunset	52,341	53,173
winaw	11,440	11,700
yahoo	4,374	4,461
Total	**219,218**	**220,466**
Time	**22.8**	**4.1**

Table 4
Streaming vs Two-Pass PWC

6.1 JBIG Comparison

The penultimate change in streaming PWC is the ability to skip over inactive portions of each scanline. However the skip mechanism is only responsible for about half of the performance gain seen in the last row of Table 4. An initial version of the streaming coder without the skip mechanism requires 7.8 seconds to encode the palette image corpus.

Another data point of note in Table 4 is that streaming PWC actually compresses CCITT black/white reference document #1 better than PWC2. In fact, over the entire set of eight CCITT documents, PWC yields 3.3% better compression than PWC2. Surprisingly, none of this difference is due to the change from an edge model in PWC2 to a pixel model in PWC. The improvement is due solely to the skip mechanism!

As the results of Table 5 show, over the entire set of eight CCITT documents, PWC is actually 1% better than sequential JBIG. The reason is that the skip mechanism obviates any need for the arithmetic coder to handle high skews. The arithmetic coder in PWC uses relatively low precision but its entire precision is dedicated to accurately representing relatively low skews.

CCITT#	JBIG	PWC
1	12,884	12,685
2	8,008	7,858
3	20,052	19,834
4	49,039	49,074
5	23,272	22,994
6	11,764	11,579
7	52,306	51,570
8	13,288	13,132
Total	**190,613**	**188,726**
Time	**9.5**	**7.2**

Table 5
CCITT Black/White Test Documents

Further, even though it uses a relatively slow arithmetic coder, PWC is 25% faster than JBIG. Of course this is only true for images where one color predominates but this condition is often met in practice. PWC is still relatively immature and it is likely that its encode speed can be improved with a better implementation.

6.2 An Expanded Performance Comparison

Many of the images in the original PWC test corpus are quite small and are not very suited for performance analysis. For this reason an expanded corpus was devised to better map PWC performance. The expanded corpus, summarized in Table 6, contains five image groups. Each group is designed to cover a specific performance regime.

Name	Description
bw	CCITT black and white facsimile documents 1-8
pc	JPEG-LS test document pc reduced to one color component
dither	Color reduced images with heavy dithering: arial and fractal
lena	USC Lena 512x512 converted to grayscale
corpus	Original PWC palette image corpus

Table 6
Performance Test Corpus

Four high performance reference coding methods, bzip[9], PNG, GIF, and JPEG-LS, were compared against PWC on the expanded corpus. Compression and performance results are summarized in Table 7 and Table 8 respectively. Since bzip, PNG, and GIF are assymetric, both encode and decode times are shown separated by a slash. Two encode times are shown for JPEG_LS. The first time is measured elapsed time. The second time is the compression time reported by the locoe program.

	PWC	bzip	PNG	GIF	JPEG-LS
ccitt	188,726	372,640	418,490	464,437	577,849
pc	104,033	198,723	225,936	376,482	361,570
dither	481,270	485,321	544,694	645,732	559,652
lena	162,478	173,645	223,051	230,921	138,883
corpus	220,426	288,363	323,290	376,208	415,734
total	**1,156,933**	**1,518,692**	**1,735,461**	**2,093,780**	**2,053,688**

Table 7
Compressed File Byte Counts

The compression results of Table 7 demonstrate that PWC operates well across a wide variety of palette images. PWC achieves the best compression on every image group excepting the grayscale representative, lena. Compression is respectable even on lena.

Even though PWC is a fairly complicated model, performance is quite reasonable compared to other encoders. However, the extremely fast decode speed of the dictionary methods appears to be unattainable by PWC-like methods.

PWC performance is relatively poor on highly dithered and grayscale images. The reason is that PWC is going through its entire decision structure and gaining only relatively modest additional compression for each decision. Table 9 shows decision counts and decisions per pixel (Dpp) for each of the example image groups. On the dither group, PWC is actually making almost 1.25 decisions for each uncompressed image bit. Perhaps surprisingly, the amount of time spent on the arithmetic multiply/divide operation never exceeds 20% of the total compression time.

	PWC	BZIP	PNG	GIF	JPEG-LS
ccitt	7.2	10.2/3.7	5.9/1.1	3.1/2.3	7.3/6.5
pc	3.5	9.2/2.5	3.8/0.6	1.9/1.4	2.2/2.0
dither	6.5	6.8/2.6	2.4/0.5	2.0/1.1	2.0/1.7
lena	2.7	2.8/1.3	1.2/0.3	0.9/0.4	0.7/0.6
corpus	4.1	7.3/2.5	2.7/1.1	2.0/1.5	3.4/2.1
total	**24.0**	**36.3/12.6**	**16.0/3.6**	**9.9/6.7**	**15.6/12.9**

Table 8
Compression Times (seconds)

Group	Decisions	Dpp	MulDiv
ccitt	4,773,463	0.15	12%
pc	2,741,487	0.80	14%
corpus	2,934,234	0.50	12%
lena	2,551,683	9.74	17%
dither	7,378,002	9.85	20%

Table 9
PWC Decision Counts

Even on the dither group, none of the PWC decisions are entirely wasted. If decisions **Q0-Q3** are turned off, Dpp drops to 8.5 but the total compressed file size increases 22%. Compression time decreases 43% and the percentage of time spent on arithmetic multiply/divide increases to 34%.

7. Analysis

PWC throughput ranges from 26.2 to 76.6 Kbytes/sec on the ccitt and dither groups respectively. Average throughput is probably close to that exhibited on the corpus group: 54.2 KB/sec. Significantly, the lowest throughput occurs when compression is greatest.

The GIF equivalent throughput, $GIF\ Size \div PWC\ Time$, has a much narrower range: 78.9 to 107.6 KB/sec for ccitt and pc respectively. This range is the performance break; a 200 MHz Intel Pentium client can expect better performance from PWC over slower connections and from GIF over faster connections.

Format	Navigator	Explorer	28.8Kb
GIF	2.5	2.0	2:03
PNG	6.0	2.0	1:51
PWC	3.5	4.0	1:13

Table 10
In Browser Performance

Table 10 shows in system performance for PWC, PNG, and GIF in an Internet browser environment. The times shown are for load and display of the palette image corpus. The first column shows local hard-drive performance under Netscape Navigator 4.x and exposes some sort of problem with the Navigator PNG implementation. The second column shows local hard-drive performance under Microsoft Internet Explorer 4.x. The last column shows IE performance over a 28.8Kbit/sec dialup connection.

7.1 Notes on Dictionary Compression

In the author's experience, the deflate compression mechanism used in PNG compresses images significantly better than the LZW method used in GIF. Decode speed for inflate is generally faster than LZW. Default encode speed is slower for deflate but can be sped up with parameter adjustment. At a given compression, deflate encodes about as fast as LZW. The "-3" setting of ZIP models LZW compression and speed fairly well.

The PNG format also allows for a flexible predictive coding mechanism called "filters"[10]. For palette images, the best filter is typically no filter. To verify this general assumption, the PNG compression parameter space was searched on the ccitt, pc, lena, and dither image groups of the previous section. The results are summarized in Table 11.

Group	Size	Delta	Filter	Level	Time
ccitt	394,888	5.7%	0	-9	27.8
pc	215,756	4.4%	0	-9	18.3
lena	150,626	32.5%	5	-4	3.5
dither	540,631	0.7%	0	-3	1.8

Table 11
PNGCRUSH Results

For the three palette image groups, no filter produced the best compression. Placing the fewest restrictions on dictionary structure yielded the best compression for ccitt and pc. However, compression times worsened significantly and compression efficiency did not markedly improve. On the dither group, restricting the dictionary to smaller entries slightly improved compression and sped things up.

Compression of lena sharply improved when using filters. The best result was achieved by adaptively selecting the optimal filter for each scanline (mode 5). Adaptation improved compression about 1.5% over exclusive use of the Paeth filter. The Paeth filter is similar to the LOCO-I predictor.

8. Notes on the Experiments

All compression times shown in the experiments were obtained on a 200 MHz Intel Processor running Microsoft Windows 95. The compression results for PWC2 were obtained using a slightly improved version of the coder described in the original PWC paper. The streaming PWC coder used in the experiments is available at http://www.netcelerate.com. PWC compression times were obtained using the "-flip" option of the coder.

JBIG results were obtained using the JBIGKIT[11]. Bzip results were obtained using bzip2[12]. JPEG-LS results were obtained using the HP Labs LOCO-I implementation[13]. PNG and GIF file sizes were obtained using Paint Shop Pro[14]. PNG compression times were simulated using command line versions of Zip[15] and Unzip[16]. GIF compression times were simulated using a PC version of UNIX compress[17]. The PNGCRUSH[18] utility was used to search the PNG compression parameter space. The arial and fractal images were obtained from Nasir Memon.

9. Summary

Streaming PWC significantly enhances the coding speed of two-pass PWC while almost completely retaining its compression efficiency. The key to the performance increase is an augmented model that reduces the average number of coded decisions. The new skip model makes coding speed largely proportional to image complexity.

The skip model has general utility in decreasing the maximum skew seen by a statistical coder. The combination of the skip model with a low precision count based binary coder out-compresses JBIG on the CCITT black/white test documents and is faster as well.

10. References

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

[2] Stephen R. Tate, "Lossless Compression of Region Edge Maps", CS–1992–9, Department of Computer Science, Duke University, Durham, NC, 1992.

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

[4] Marcelo J. Weinberger, Gadiel Seroussi, and Guillermo Sapiro, "LOCO-I: A low Complexity, Context-Based, Lossless Image Compression Algorithm", *Proceedings Data Compression Conference*, March 1996, IEEE Press, Los Alamitos, California.

[5] Don Speck, Carry-free arithmetic coder, personal communication, April, 1996.

[6] Don Speck, "Arithmetic coder combining the best compression with the best speed", ISO/IEC JTC1/SC29/WG1 submission, July 11, 1997.

[7] Lèon Bottou, Paul G. Howard, and Yoshua Bengio, "The Z-Coder Adaptive Binary Coder", *Proceedings Data Compression Conference*, March 1998, IEEE Press, Los Alamitos, California.

[8] 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).

[9] M. Burrows and D.J. Wheeler, "A Block-sorting Lossless Data Compression Algorithm", SRC Research Report, Digital Systems Research Center, 130 Lytton Avenue, Palo Alto, California 94301.

[10] PNG (Portable Network Graphics) Specification, http://www.w3.org/TR/REC-png.

[11] 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.

[12] Julian Seward, bzip2, http://www.muraroa.demon.co.uk

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

[14] Jasc, Inc., http://www.jasc.com.

[15] 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.

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

[17] WinXs Unix Tools for Windows, available for download at the ZDNet Software Library, http://www.hotfiles.com.

[18] Glenn Randers-Pehrson, *pngcrush*, http://www.netgsi.com/~glennrp/pngcrush.