Circular Coding for Data Embedding

Robert Ulichney, Hewlett-Packard Co., Andover, MA, USA
Matthew Gaubatz, Hewlett-Packard Co., Seattle, WA, USA
Steven Simske, Hewlett-Packard Co., Ft. Collins, CO, USA

Abstract

A general two-dimensional coding means is presented that allows recovery of data with only a cropped portion of the code, and without knowledge of the carrier image. A description of both an encoding and recovery system is provided, along with an illustrative example. Our solution involves repeating a payload with a fixed number of bits assigning one bit to every symbol in the image, whether that symbol is data carrying or non-data carrying, with the goal of guaranteeing recovery of all the bits in the payload. The system uses row-to-row offset itself to communicate the value of the phase of the circular payload. The recovery system is given the number bits in the payload, evaluates each candidate shift and ranks its confidence based on the variance of the payload bits. Symbols determined to be unsuitable for recovery are labeled "abstentions" and not included in the decoding process; special consideration is given to the checkerboard subsampling that can occur in the case when halftone cells are used as the data-carrying symbols. This particular application is examined via tests to quantify the likelihood of unrecoverable bits and bit redundancy as a function of phase and crop window size.

Introduction

Embedding data in hardcopy is increasingly important for content linking, security and other applications. Data-bearing hardcopy is most often accomplished with various types of multidimensional barcodes, along with more aesthetically pleasing alternatives of encoding symbols in halftones [1][2]. We propose a new solution for recovering data when only a part of the data-bearing image can be captured, as in Figure 1, with no knowledge of the carrier image. The solution is designed to be robust in the presence of one or more of the many print-scan process degradations [3] that can hurt data integrity.

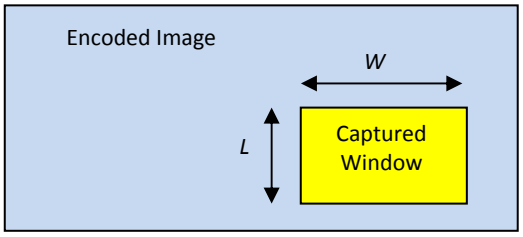

Figure 1.Image and Capture Region.

The following notation is used to describe the approach:

- B Bit count of Payload, known to encoder and decoder.
- **P** Payload bits, representing a value between 0 and 2^B-1.
- D Row-to-row Offset of the payload in the image.
- **P'** Candidate recovered Payload.
- D' Candidate row-to-row offset.
- **S** Payload in Standard form.
- C Circular shift to get the payload from the standard form.
- L Length of the crop window.
- W Width of the crop window.

Encoding System

The goal is to represent a payload, **P**, consisting of B bits in a two-dimensional array of symbols. One bit is assigned to every symbol in the image, whether that symbol is data carrying or non-data carrying. The size of the array is W symbols in width by L symbols in length. For each row of symbols, the payload is repeated until the end of the row. For each successive row the payload is circularly shifted D bits relative to the row above it.

Circular Shifting and Standard Form

When cropping a pattern of a repeating sequence of bits it is important to point out that the individual bit positions within the cropped content can be ambiguous. However, any contiguous sequence of B bits will represent one of B circularly shifted versions of that sequence. As a very simple 4-bit example, the table below illustrates the different versions of an example 4-bit code 1100_2, where "Shift" is the number of bits the code is circularly right shifted.

Shift	b_3	b_2	b_1	b_0	Decimal Value
0	1	1	0	0	12
1	0	1	1	0	6
2	0	0	1	1	3
3	1	0	0	1	9

The original code is completely determined if both the Shift value and bit sequence are known. By defining a differentiating criterion, any circularly shifted code can have one version that is always singled out from the other versions. One such criterion is based on the value of the shifted version. In the above table, note that a shift of 2 yields the smallest value and a shift of 0 yields the largest value. The Standard Phase of a Payload is represented as **S**, and follows the convention that the Standard Phase corresponds to the Circular Shift C yielding the smallest version of the shifted code. In our example, $\mathbf{P}=1100_2$, $\mathbf{S}=0011_2$ and $C=2$.

Note that the shifted versions of some Payloads may have the same value. $\mathbf{P}=1010_2$ is one such example. Note that the Standard Phase $\mathbf{S}=0101_2$ can be achieve with a Circular Shift $C=1$ or $C=3$. As the result is identical it does not matter which one is used. A successful encoding scheme enables recovery of (1) all of the bits in the circular payload, and (2) the value of the circular shift C.

Encoding Phase in the Offset

Encoding is achieved by repeating a stream of codes **S** in a two-dimensional array where each successive row is circularly shifted by D samples with respect to row above, where D is an integer multiple of C, i.e., $D=kC$, where k is an integer. Since our payload is B bits, the accumulated shift can be identically expressed by reducing it by modulo-B.

A detailed example for the payload $\mathbf{P}=1100_2$ is discussed. While this code is very short (B=4) it exemplifies the same process performed on longer payloads. Figure 2 illustrates this encoding in an 11 by 21 array of one-bit symbols. As was shown earlier, $\mathbf{P}=1100_2$ has a Standard Phase $\mathbf{S}=0011_2$ and Circular Shift C=2. In this simple example the row-to-row shift $D = C = 2$, that is, k=1. The first row starts with a zero-shifted version of **S** repeated across all symbols in the row. The second row is circularly right shifted by D=2. The third row shifts the second row by D=2 or the first (standard form) row by D=2+2=4, or the simpler Modulo-4 shift of 0. The pattern continus for all lines completing the two-dimensional array. A cropped subset of size W by L (= 9 by 6) is depicted by the rectangular black outline.

Shift	Mod *B*																				
0	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
2	2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
4	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
6	2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
8	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
10	2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
12	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
14	2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
16	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
18	2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
20	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

Figure 2. Example encoding of $\mathbf{P}=1100_2$, $\mathbf{S}=0011_2$ in an array of symbols. Black frame indicates captured crop region.

Recovery System

The capture device only reads a cropped W x L subset of the array. The number of payload bits B and the offset-to-shift integer k must be communicated to the recovery system, the purpose of which is to find the Payload **P**. Recovery is achieved through the following five steps.

(1) Decode each symbol in the cropped region, and assign each a value of 0, 1 or "abstain". If the symbol is degraded due to damage, noise, or otherwise is deemed un-decodable it will not be included in the recovery process.

(2) For each candidate row-to-row offset $D'_i \in \{0, k, 2k, \ldots, k(B\text{-}1)\}$, find an estimate of a shifted payload $\mathbf{P'}_i$, and the confidence associated with that estimate. Given a candidate shift, for each bit position $\{b_{B\text{-}1}, \ldots, b_1, b_0\}$ find an average of all non-abstained values and assign a value of 0 if that average is less than 0.5 and a value of 1 if that value is greater than or equal to 0.5. The uncertainty, u_j, for each bit position is then the absolute value of the difference between bit estimate and the average. A bit's uncertainty will range from 0 to 0.5. The *Confidence* of a candidate shift is then

$$Confidence = 1 - (2/B)\Sigma u_j, \text{ for } j=\{0, 1, 2, \ldots, (B\text{-}1)\},$$

and ranges from 0 to 1.0.

(3) Select the value of D' and **P'** with the highest value of *Confidence*.

(4) Convert **P'** into Standard Phase **S** by finding the minimum value of all B circular shifts of **P'**.

(5) Circularly right shift **S** by $C=D/k$ to obtain the Payload **P**.

Example

To continue the with the values used in the encoding example above, the crop used for recovery is shown as the 9x6 boxed subset of symbols in Figure 2.

(1) To simplify the recovery example, it is assumed that there are no abstentions and each symbol is decoded as a 0 or 1 as shown.

(2) The value B=4 is available to the recovery system, so there are 4 candidate shifts $D' = \{0,1,2,3\}$ to test. The process of testing one candidate shift, $D'=1$, is shown in Figure 3. For this Candidate Shift the 9x6 crop of decoded symbols is shown with the leftmost bit position of the shifted payload highlighted in yellow to aid in following the process. "Samples" indicates the number of values for each of the four bit positions; since there are 9x6 total values, the sum of these values from each case is 54. "Sum" represents the sum of all 0 and 1 values for each bit position, and "Average" is simply Sum/Samples. This Average is then rounded to find each bit in **P'**. Uncertainty is the absolute value of the difference between the average and the rounded value. Finally, Confidence is calculated as described above. This same process is repeated for each of the other candidate shifts.

	B_3	b_2	b_1	b_0	Data with b_3 highlighted
Samples	14	13	13	14	0 1 1 0 0 1 1 0 0
Sum	4	7	8	8	1 0 0 1 1 0 0 1 1
Average	0.29	0.54	0.62	0.57	0 1 1 0 0 1 1 0 0
P'_1	0	1	1	1	1 0 0 1 1 0 0 1 1
Uncertainty	0.29	0.46	0.38	0.43	0 1 1 0 0 1 1 0 0
					1 0 0 1 1 0 0 1 1

Figure 3 . Recovery process example including the computations associated with evaluating candidate shift =1. The candidate payload 0111_2 has a confidence of only 22% [=1-(2/4)(.29+.46+.38+.43)].

(3) The Candidate row-to-row offset D' with the maximum Confidence is D'=2. The corresponding Shifted Payload is then 0110_2. Since k=1, C=D=2.

(4) The Standard Phase of 0110_2 is $\mathbf{S}=0011_2$.

(5) Circularly Right Shifting 0011_2 by C=2 correctly delivers the Payload $\mathbf{P}=1100_2$.

Modification for Halftone Encoding/Recovery

To use this form of circular coding, stegatone [1] encoding is modified so that every cell is treated like a 1-bit carrier and payload bits are allocated accordingly. Stegatones are clustered-dot halftones meaning that at most only half of the cells can successfully carry data. This constraint is due to the fact that in any highlight region half of the cells are completely white and in any shadow region half of the cells are completely black; solid

black and white cells are incapable of shifting or carrying data. Thus, at least half of the cells will be interpreted as "abstentions" in the recovery process.

Figure 4 illustrates a zoomed and cropped region of an image halftoned for stegatone encoding. The 23x23 cells shown span a region of shadow (left) and highlight (right). The red dots indentify the shadow cell checkerboard; while their positions can be data bearing in the shadow region they are void of data in the highlight region because they are all white.

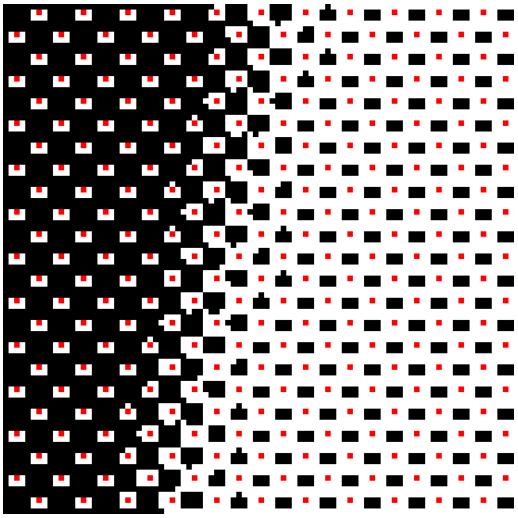

Figure 4. Zoomed halftone image with shadow cells indicated by red dots.

A key issue to be addressed is that the captured crop will very often be an entirely shadow or an entirely highlight region. In such regions, every cell on a checkerboard arrangement will be designated as an abstention.

18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	63	62	61	60
59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37
36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14
13	12	11	10	9	8	7	6	5	4	3	2	1	0	63	62	61	60	59	58	57	56	55
54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
8	7	6	5	4	3	2	1	0	63	62	61	60	59	58	57	56	55	54	53	52	51	50
49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27
26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4
3	2	1	0	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45
44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	63
62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40
39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	63	62	61	60	59	58
57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35
34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
11	10	9	8	7	6	5	4	3	2	1	0	63	62	61	60	59	58	57	56	55	54	53
52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30
-29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
6	5	4	3	2	1	0	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2

Figure 5. Bit positions indicated in a 23x23 capture of a B= 64 bit payload with offset D=41.

Consider the example of a 23x23 crop of clustered-dot halftone cells shown in Figure 5; the numbers are bit positions with gray squares representing highlight cells and white squares representing shadow cells. In this example, with B=64 and D=41, half of the data bits would be unrecoverable in an area of all highlight or all shadow, as all even bit positions fall on highlight cells and all odd bit positions fall on shadow cells. This behavior will occur whenever B is even and D is odd.

As will be shown, this problem is avoided by setting k=2 such that D=2C will always be even for the case that B is even. To continue the example, consider the 23x23 crop with B=65 and D=2 in Figure 6. Even if only highlight (gray) or shadow (white) cells are present, all 65 bit positions are represented.

55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35
59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37
61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41
0	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43
2	1	0	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45
4	3	2	1	0	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47
6	5	4	3	2	1	0	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
8	7	6	5	4	3	2	1	0	64	63	62	61	60	59	58	57	56	55	54	53	52	51
10	9	8	7	6	5	4	3	2	1	0	64	63	62	61	60	59	58	57	56	55	54	53
12	11	10	9	8	7	6	5	4	3	2	1	0	64	63	62	61	60	59	58	57	56	55
14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	64	63	62	61	60	59	58	57
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	64	63	62	61	60	59
18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	64	63	62	61
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	64	63
22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4
28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12

Figure 6. Bit positions indicated in an example 23x23 capture of a B= 65 bit payload with offset D=2.

Analysis

It is important to understand the complex interaction between quantities that are affected by crop window size, for all possible circular shifts and even or odd payload bit sizes, specifically to help design appropriate error correction coding strategies. For a given crop window size, it is possible to exhaustively simulate all possible crop positions within a halftone checkerboard subsampled array of symbols. Both the number of missing payload bits and the number of times payload bits are repeated can be computed.

It is also necessary to test all possible circular shifts, C. For a B-bit payload, there are B possible values C to shift codes to the standard form. While all values of C are possible, it is also important to understand the achievable distributions of values. For the example of B=10, the graph on the left of Figure 7 shows a distribution of the 2^{10} = 1024 codes with each value of C from 0 to 9 (=B-1). The data show that the distribution of numbers of codes is quite uniform; it ranges between 99 and 107 thus varying only about 8% from the mean. The reason for the slight bias for smaller values of C is likely due to the fact that for codes where more than

one value of C will shift it to the standard form, the smaller value was recorded. This non-uniformity is plotted for small values of B in the graph on the right of Figure 7, revealing that it becomes vanishingly small as B increases. Therefore all values of C carry approximately equal weight.

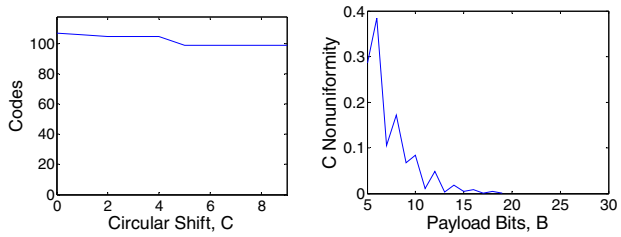

Figure 7. Uniformity testing of Circular Shifts C. Distribution for B=10 (left), and Relative Non-uniformity as a function of B (right).

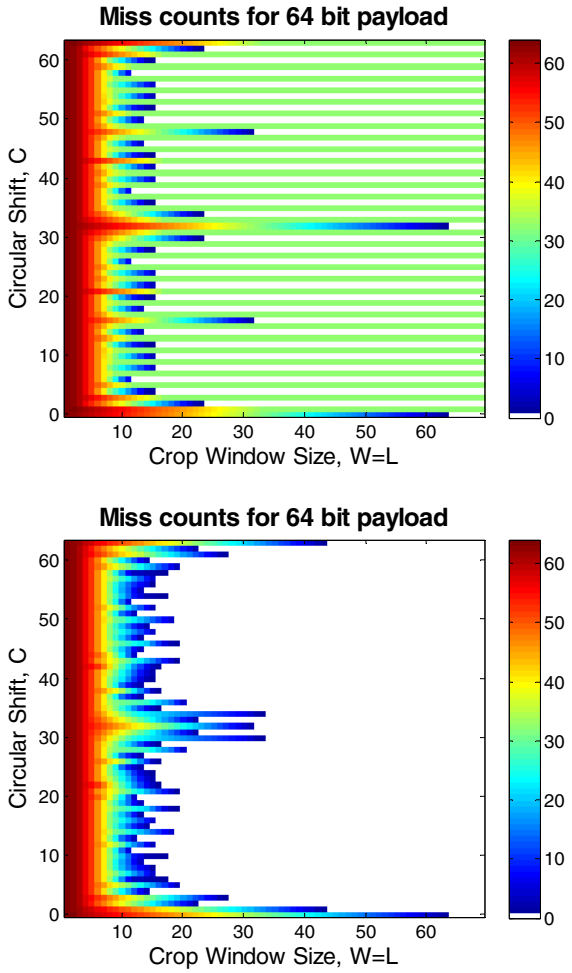

Figure 8. Worst case missed bits as function of crop window and circular shift, from an example where the number of payload bits is even, B=64. Results for an all highlight or all shadow region (top), and half highlight, half shadow region (bottom).

Counting Missing Bits

In the example of Figure 5 it is shown that with the checkerboard subsampling of an all-highlight or all-shadow halftone area, with B=64 and D=41, half the bits are unrecoverable. For this value of B, all values of circular shifts C, from 0 to (B-1)=63, are examined. In this example, k=1 and D=C. Using the method of encoding described earlier, a message was encoded in a large array for each value of C; the array was decimated by checkerboard subsampling as would be the case for an all highlight or all shadow stegatone.

As a first test, the array was cropped with square windows from sizes W=L=1 to 69. For each fixed crop size, the crop window was repeatedly shifted in single cell steps across the array. For each shifted crop window, the number of missing bits was recorded. The largest number of missing bits for the worst-case shifted crop was then plotted in the upper graph of Figure 8. In this plot the miss count is color coded as indicated. Note that the case of zero missing bits is color-coded with white. As expected, all odd values of C results in at least half of the bits missing.

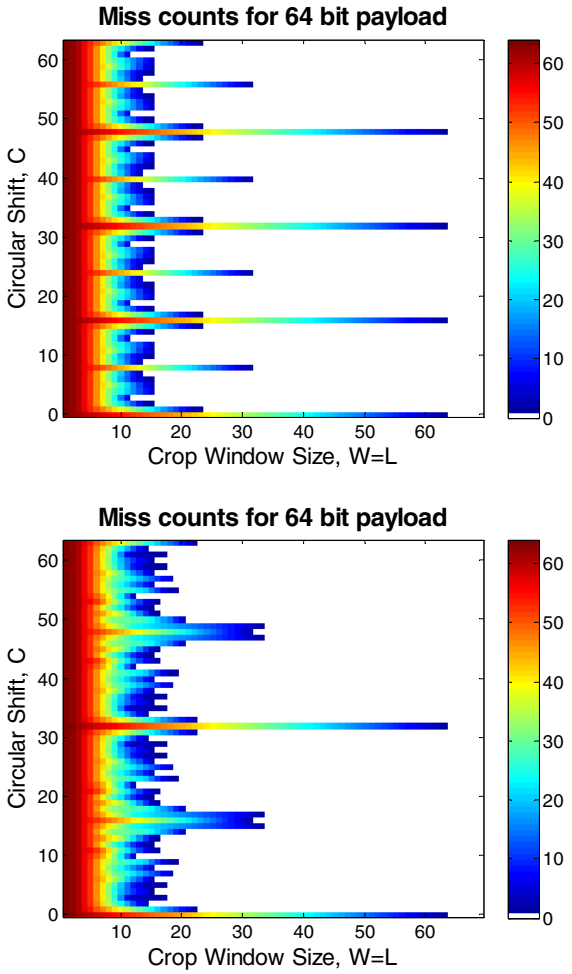

Figure 9. Worst case missed bits as function of crop window and circular shift. The B=64 example of Figure 8 for a payload with an even number of bits, but forcing the offset to be even by setting k=2 so D=2C. Results for an all highlight or all shadow region (top), and half highlight, half shadow region (bottom).

On the lower graph in Figure 8 the experiment is repeated with one change: instead of all highlight or all shadow halftone subsampling, the region examined is half highlight and half shadow. For this underlying pattern the chronic missing bit problem for odd circular shifts was eliminated. While this possibility is theoretically interesting, restricting crop windows to straddle highlight-shadow boundaries is not practical.

Since the problem is with the odd offsets, the solution is to set the value of k=2 so D=2C, rendering the offset D even for all values of C. The plots for missed bits using k=2 is shown in Figure 9. As with Figure 8, the top plot shows the results for an all highlight or all shadow region and the bottom plot for a half highlight half shadow region. Both cases are free of the chronic missing bit problem but do show the expected "compressed" pattern relative to the case in Figure 8.

Figure 10 examines B=65, a case where the number of payload bits is odd. As in the preview figures, the top plot shows the worst case missing bits results for an all highlight or all shadow region and the bottom plot for a half highlight half shadow region.

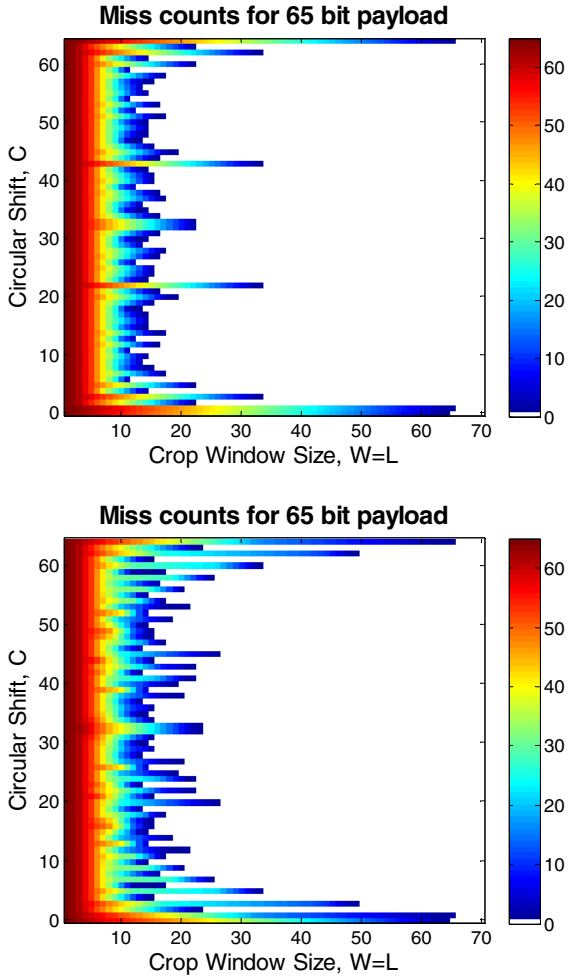

Figure 10. Worst case missed bits as function of crop window (W=L) and phase, C. Example where the number of payload bits is odd, B=65. Results for an all highlight or all shadow region (top), and half highlight, half shadow region (bottom).

In both Figure 8 and 10, the worst case number of missing bits is B for a crop window size of 1, then for any given value of C the number decreases essentially linearly down to zero missed bits.

Bit Repeat Count

While it is important to know what combinations of crop window size and shift C will allow all bits to be represented at least one time after halftone checkerboard subsampling, it is equally important to know the number of times each payload bit is represented in that window. The reason for this test is that in the presence of noise in the print and scan process, the accuracy of correctly recovering any single bit is not perfect, so redundancy is needed. To this end, the repeat counts are also presented as a function of crop window size and circular shift.

By way of example, consider the cropped window of Figure 6. Bit position 1 is repeated 6 times if the region is all shadow cells, and 5 times if the region is all highlight cells. Bit position 33 only occurs once for each case, however. The repeat count for all individual code bits over several shifted crop windows is shown in Table 1.

Table 1.

Repeats	Code Bits	Acc. Bits	Acc. Rate
8	0	0	0%
7	0	0	0%
6	1267	1267	30%
5	878	2145	51%
4	520	2665	63%
3	520	3185	75%
2	520	3705	88%
1	520	4225	100%
0	0	-	-

First, it is important to note that there are no code bits with 0 repeats, that is, no missed bits. The accumulated bits column indicates the number of code bits that have at least the number of repeats indicated, and the accumulated rate reflects the percent of code bits with at least that repeat count. What is important for code recovery in some instances is not the average bit repeat count, but the minimum repeat count of some percentage of the all code bits. The value 90% is used in this study. The example of Figure 6, where B=65, C=2, W=L=23, has a 90% repeat count of 1. A reasonable 90% repeat count would be 4 or higher, so this crop window would not be acceptable.

Figure 11 illustrates 90% repeat count values for two of the cases examined earlier for miss counts. The results for B=65, k=1 and B=64, k=2 are shown in the upper and lower graphs, respectively. In both cases, the crops are in an entirely highlight or shadow halftone area. Note how the non-zero patterns are inverted from the miss count plots.

Concluding Remarks

A system was presented for encoding and decoding a cropped payload that survives checkerboard subsampling, and the effect crop window size has on payload bit misses and repeats was quantified. For crop windows larger than B, it is clear that 90% repeat counts are very likely in situations where all bits are

recoverable. For crop windows smaller than *B*, error correction coding schemes can be tailored to the statistics shown here to accommodate some level of missing bits. The raw recovery rates for data encoded in clustered-dot halftone cells (stegatones) have been measured for a wide variety of printers and resolutions [4].

Another solution for small crop window sizes is to pad "1" bits to the raw payload to avoid *C* values with the highest miss counts. For the case of odd values of *B*, where *B*=65 illustrated in this paper is but one example, the three most troublesome *C* values are *C*=0, 1, and (*B*-1). Padding a 1 bit on the MSB side of the payload code eliminates *C*=0 form occurring. Padding a second 1 bit eliminates *C*=(*B*-1). By padding a 1 bit on the LSB side of the code eliminates *C*=1 from occurring.

The system presented here allows for blind recovery applications of steganographic halftones, as well as for other mechanisms for storing data in two dimensions. Next steps will involve applying results of this study to improve robustness of recovery.

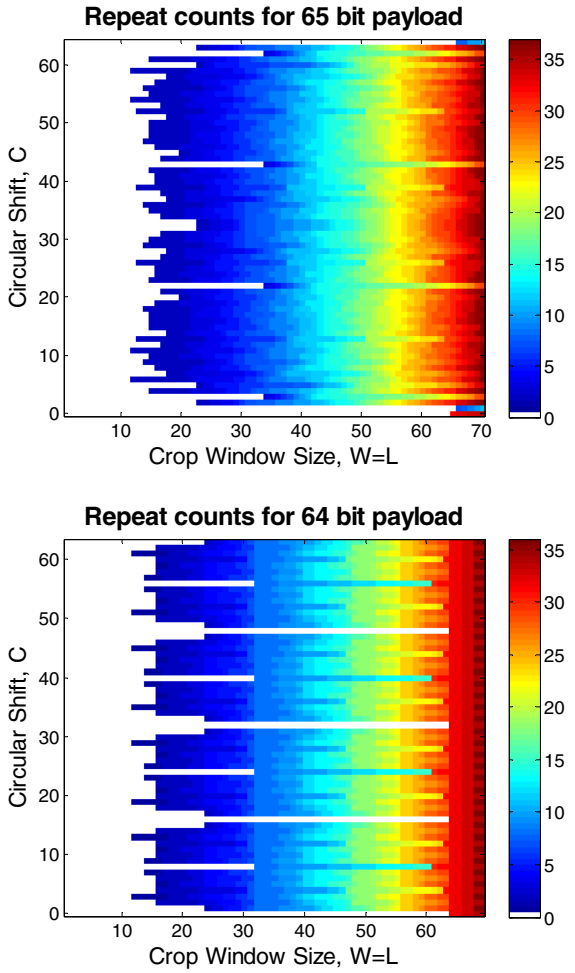

Figure 11. The 90% repeat counts for each bit in the payload as function of crop window and circular shift. The values indicate that 90% of the bits will be repeated at least that number of times. Results are shown for an all highlight or all shadow region. B=65, k=1 (top), B=64, k=2 (bottom).

Author Biography

Robert Ulichney is a Distinguished Technologist with HP Labs. He received a Ph.D. from MIT in electrical engineering and computer science. Before joining HP he was with Digital Equipment Corp for several years then with Compaq's Cambridge Research Lab where he led a number of research projects on image and video implementations for both hard copy and display products.

References

[1] R. Ulichney, M. Gaubatz, and S. Simske, "Encoding Information in Clustered-Dot Halftones", IS&T NIP26 (26th Int. Conf. on Digital Printing Technologies), Austin, TX, 602-605, Sep 2010.

[2] O. Bulan, G. Sharma, and V. Monga, "Orientation Modulation for Data Hiding in Clustered-Dot Halftone Prints," IEEE Transactions on Image Processing, vol. 19, no. 8, pp. 2070-2084, Aug 2010.

[3] K. Solanki, U. Madhow, B. S. Manjanuath, S. Chandrasekaran and I. El-Khalil, "'Print and Scan' Resilient Data Hiding in Images," IEEE Transaction on Information Forensics and Security, vol. 1, pp. 464-478, Dec 2006.

[4] Y. Chen, R. Ulichney M. Gaubatz, S. Pollard, "Stegatone Performance Characterization", *Media Watermarking, Security, and Forensics 2013*, IS&T/SPIE Electronic Imaging Symposium, San Francisco Airport, CA, 8665-27, Feb 3-7, 2013.