Fused Number Representation Systems and their Barcode Applications

Sarkis Agaian*
Department of Electrical Engineering, Stanford University,
450 Serra Mall, Stanford, California 94309

ABSTRACT

In this paper we focus on: a) enhancing the performance of existing barcode systems and b) building a barcode system for mobile applications. First we introduce a new concept of generating a parametric number representation system by fusing a number of representation systems that use multiplication, addition, and other operations. Second we show how one can generate a secure, reliable, and high capacity color barcode by using the fused system. The representation, symbols, and colors may be used as encryption keys that can be encoded into barcodes, thus eliminating the direct dependence on cryptographic techniques. To supply an extra layer of security, the fused system also allows one to encrypt given data using different types of encryption methods. In addition, this fused system can be used to improve image processing applications and cryptography.

Keywords: Barcode, Multiple Base Representation, Fibonacci/Lucas p-codes, Double Base Representation, Cell/ mobile phones

1. INTRODUCTION

Since the 1960's, barcodes have provided optical machine-readable representations of data. Initially used to label railroad cars, barcodes are now laid on almost every medium, from paper and cardboard to plastics and metals and from skin to DNA[1]. Though other technologies are being developed for consumers to scan objects, including radio waves, computer chips or satellite location systems, barcodes maintain their popularity due to their low cost, low rate of error, high information capacity, and portability[2]. The potential and demand for barcode technology continues to increase with constantly improving digital technologies. The recent integration of these technologies, namely through the pairing of mobile camera phones and high density barcodes, can fulfill the demand for electronic data exchange. For instance, in Japan and Korea, people can save and exchange Facebook information, receive the nutritional information about their McDonald's meal, board airplanes using their phones and wireless payments[3]. Despite its huge potential, however, the use of camera phones to scan barcodes is not widespread. Resolution limits, distortion, blurring, and noise induced by the phone camera hinder the direct use of most existing bar codes for mobile phones[4]. These problems are exacerbated when dealing with camera phones that generate low to medium quality images (less than 640 x480 pixels)[5]. Questions that have been raised include: Are the existing barcodes suitable for scanning by mobile phones? Is it necessary to design a new barcode tailored for mobile phones[4]? As private wireless transactions may be prone to hacking and piracy, an additional question is how can make barcodes more secure?

Companies have developed their own unique, proprietary barcode formats that are better geared toward mobile applications[2]. These developments have resulted in designs that vary dramatically from those of traditional barcodes, which represent data in the widths and spacings of black parallel lines. These barcodes, also known as linear or 1D (1 dimensional) barcodes or symbologies carry a limited amount of information with few security features. Other

*sagaian@stanford.edu

Multimedia on Mobile Devices 2010, edited by Reiner Creutzburg, David Akopian, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7542, 754207 · © 2010 SPIE-IS&T · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.839111

symbologies, such as 2-D and color barcodes, have thus been looked at in an effort to make barcode systems more efficient, accessible, and secure. Many of these symbologies are similar to their linear counterparts in that they are based on binary representations. They differ, however, in the amount of information they can hold. In exchange for the increased information content, 2D and color barcodes have also increased the complexity of recognition algorithms, as they require greater image segmentation[6]. Thus, due to the varying resolutions of mobile camera phones, the imaging quality of a given camera phone may not be satisfactory for decoding a high density 2D or color barcode[4]. In addition, image capture by the phone camera in a variety of lighting conditions may not retain enough discriminate information of color and grayscale.

In the following, we introduce a new representation system to generate secure, reliable, and high capacity barcodes that are easier for cell phones to decode. We call this representation system the Multiple Base Number System (MNBS). MBNS fuses several number representation systems including the double base number system, which has recently been introduced to offer more sparse and efficient means of representing data. MBNS can be applied to barcode symbols with a variety of widths, colors, and vertical arrays to represent more information. The rest of this paper is organized as follows: Section II introduces some necessary background information. A concept of generating fused number representation system is provided in Section III. In Section IV, the fused number representation system is used to construct more efficient and error-resistant barcode systems. Section V offers concluding remarks and suggestions for future work.

2. BACKGROUND: NUMBER REPRESENTATIONS

Below we review some number representations, most notably Fibonacci and Lucas numbers and their generalizations. We also introduce new Fibonacci and Lucas p-number based parametric representation number systems, which are later implemented in developing color barcodes.

Fibonacci and Lucas Sequences: Fibonacci and Lucas (F-L) sequences include weighted F-L numbers, F-L p-numbers, n-dimensional (Meta) F-L numbers, and random F-L numbers. In this article, we focus on sequences that are generated using the first two classes of numbers.

Fibonacci and Lucas (F-L) Numbers[7]: Fibonacci (f_n) and Lucas (l_n) numbers can be presented recursively as

$$f_k = f_{k-1} + f_{k-2}, \quad l_k = l_{k-1} + l_{k-2}, \tag{1}$$

where $f_0=0$, $f_1=1$, $l_0=2$, $l_1=1$, $l_2=3$, $k \geq 2$.Subsequent terms of F-L numbers are thus defined as the sum of their two predecessors. Note that a Lucas number l_k can be also expressed in terms of the Fibonacci numbers as:

$$l_k = 2f_k + f_{k-1};\ l_k = f_{k-1} + f_{k+1};\ f_k = (l_{k-1} + l_{k+1})/5, \tag{2}$$

Weighted Fibonacci and Lucas (F-L) P-Numbers[7]*:* Numbers that are given "weight" based on their coefficients and are defined by the following recurrence:

$$F_n^{(p)} = \begin{cases} 0 & if\ n \leq 0 \\ 1 & if\ \ 0 < n \leq p+1 \\ aF_{n-1}^{(p)} + bF_{n-p-1}^{(p)} & \text{if} \quad n>p+1 \end{cases}, \tag{3}$$

$$L_n^{(p)} = \begin{cases} p+1 & if\ n = 0 \\ 1 & if\ \ 0 < n \leq p \\ aL_{n-1}^{(p)} + bL_{n-p-1}^{(p)} & \text{if} \quad n>p \end{cases}, \tag{4}$$

where a and b are some constants and p is a non-negative integer. In the following, we consider the case in which a = b = 1. Note that F-Lucas p-number representations include an "infinite" number of various representations. For

values p=0 and p=1, for instance, the F-L p-representations respectively reduce to the well-known binary and F-L systems previously discussed. As $p \to \infty$, the F-L p-representations become the "unitary code" in which $X = \underbrace{1+1+...1}_{X}$ Other number systems can be generated by varying the value of p. The initial sequences for the first five p-values are given in the table below.

Table of Fibonacci and Lucas P-Numbers

P	Fibonacci p-numbers	Lucas p-numbers
0	0,1,2,4,8,16,32,64,128,512,1024…	1,1,2,4,8,16,32, 64,128,256,512,1024, 2048…
1	0,1,1,2,3,5,8,13,21,34,55,89,143…	2,1,3,4,7,11,18,29,47,76, 123, 199, 322…
2	0,1,1,1,2,3,4,6,9,13,19,28,41,60…	3,1,1,4,5,6,10,15,21,31,46, 67, 98…
3	0,1,1,1,1,2,3,4,5,7,10,14,19,26,36,50,69,95..	4,1,1,1,5,6,7,8,13,19,26, 34, 47…
4	0,1,1,1,1,1,2,3,4,5,6,8,11,15,20,26,34,45,60,80…	5,1,1,1,1,6,7,8,9,10,16,23, 31…

Parametric Representations: Consider the following expansion of an integer X

$$X = a_k R(k,u) + a_{k-1} R(k-1,u) + a_{k-2} R(k-2,u) + ... + a_2 R(2,u) + a_1 R(1,u) + a_0, \quad a_m \in Z \qquad (5)$$

where $R(k,u)$ is a parametric base sequence of the number system. The table below gives some sample representation systems we can generate from (5).

Table of Parametric Representations

Case	R(k,u)	Representation
Signed Binary	2^k[8]	$a_k 2^k + a_{k-1} 2^{k-1} + a_{k-2} 2^{k-2} + ... + a_2 2^2 + a_1 2^1 + a_0, \quad a_m \in \{-1,0,1\}$ (6)
Signed Ternary	3^k[9]	$b_k 3^k + b_{k-1} 3^{k-1} + b_{k-2} 3^{k-2} + ... + b_2 3^2 + b_1 3^1 + b_0, \quad b_m \in \{-1,0,1\}$ or $\{0,1,2\}$ (7)
Signed General	r^k[10]	$c_k r^k + c_{k-1} r^{k-1} + c_{k-2} r^{k-2} + ... + c_2 r^2 + c_1 r^1 + c_0, \quad c_m \in \{0, \pm 1, \pm 2, ..., \pm(r-1)\}$ (8)
P-Fibonacci	$F^{(p)}_k$[7]	$b_k F^{(p)}_k + b_{k-1} F^{(p)}_{k-1} + ... + b_2 F^{(p)}_2 + b_1 F^{(p)}_1, \quad b_m \in \{0,1\}$, $F^{(p)}_m$ is a Fibonacci p-number $\le X$ (9)
Double Base	$p^k q^u$[11]	$d_{k,u} p^k q^u + d_{k-1,u-1} p^{k-1} q^{u-1} + d_{k-2,u-2} p^{k-2} q^{u-2} + ... + d_{0,0}, \quad d_{m,n} \in \{0,1\}$, (10) $(generalized)$ $k < u$, $0 \le d_{m,n} < k$; $(specifically) \quad k = 2, u = 3$
Prime	P_k[12]	$e_k P_k + e_{k-1} P_{k-1} + ... + e_2 P_2 + e P_1, \quad e_m \in \{0,1\}$, P_m is a prime number $\le X$ (11)

Many such systems are sparse but redundant. We are interested the conditions that lead to these systems' unique representations and minimal expansions. For instance, it has been shown that the representation R(k,u) = q^k, where q is an integer ≥ 2 is unique and minimal if two consecutive digits $c_k c_{k-1} = 0$ for all k [8], or the coefficients $c_m, \quad m = 0,1,2,...,k$ satisfy the following two conditions[13]:

$$\begin{cases} |c_m + c_{m+1}| < r, & for \quad all \quad m \\ |c_m| < |c_{m+1}|, & if \quad c_m c_{m+1} < 0 \end{cases},$$

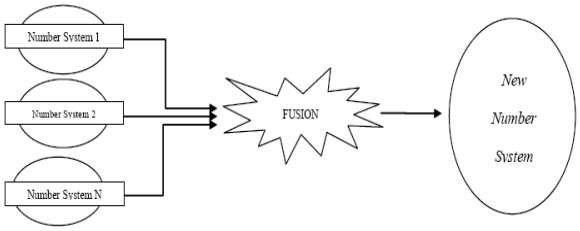
Case 1 can also be extended to finding a unique Fibonacci representation (Zeckendorf's Theorem). Such a representation for an integer X exists if: $X = F_k - 1$, k=1,2,3,4…Both Fibonacci and q^k representations have been applied in the optimal design of arithmetical hardware[8], coding theory[6], and cryptography[11,14].

3. FUSED NUMBER REPRESENTATION SYSTEMS

In this section we introduce the Multiple Base Number System (MBNS), a collection or fusion of different number systems. MBNS allows us to represent an integer X with various systems $\left\{S_n^{(p)}, T_m^{(q)},...,Z_k^{(l)}\right\}$ in the form

$$X = \sum_{i,j,..,k} s_{i,j} S_n^{(p)} * T_m^{(q)} * ... * Z_k^{(l)}, \quad p,q,l = 0,1,2,... \qquad (12)$$

where * is an arbitrarily chosen "fusion" operation (i.e., addition, multiplication, linear combination, etc.) and $s_{i,j}$ is either real or imaginary. Some noteworthy s_{ij} values include $s_{i,j} \in \{0,1\}$, $s_{i,j} \in \{-1,0,1\}$, $s_{i,j} \in \{0,1,j\}$ where $j = \sqrt{-1}$. Also to be noted is that $S_m^{(p)}, T_n^{(q)},...,Z_k^{(1)}$ can be both real and complex number systems. Changing the parameter values in these systems provides greater representation possibilities. The map below sums up the basic idea behind this new number system.



In the following, we focus on a subset of MBNS, namely the two base representation system which allows us to fuse two number systems together. The ideas below, however, can be extended in fusing multiple other number systems.

Two Base Representation System: A subset of MBNS in which two number systems, $S_m^{(p)}, T_n^{(q)}$, are fused into a single parametric representation of the form:

$$X = \sum_{i,j} e_{i,j} S_m^{(p)} * T_n^{(q)}, \quad p,q = 0,1,2... \qquad (13)$$

$e_{i,j} \in \{0,1\}$, *or* $e_{i,j} \in \{-1,0,1\}$, $e_{i,j} \in D = \{0,...,|q|\}, q = 1,2,...$ $e_{i,j} \in \{0,1,j,j+1\}$ $j = \sqrt{-1}$ *etc.*

where n, m ∈ {0,1,2,3…} and p and q are parameters. Examples of (13), in which we let * be the multiplication operation, are provided below to illustrate the form's versatility.

- $S_m^p = 2^m, T_n^q = 1, \quad e_{i,j} \in \{0,1\}$, gives the radix-2 representation (see 6);
- $S_m^p = 3^m, T_n^q = 1, \quad e_{i,j} \in \{0,1,2\}$, gives the radix-3 representation (see 7);
- $S_m^p = 2^n = T_n^q, \quad e_{i,j} \in \{0,1\}$, gives the radix -4 representation;
- $S_m^p = 2^m$ or $3^m, T_n^q = 1, \quad e_{i,j} \in \{-1,0,1\}$, gives the signed–digit number system (see 6,7);
- $S_m^p = 2^m, T_n^q = 3^n, \quad e_{i,j} \in \{0,1\}$, gives the double-base representation (see 10);
- $S_m^p = F_m^p$ or $L_m^p, T_n^q = 1, \quad e_{i,j} \in \{0,1\}$, gives the F-L p-representation(see 9);
- $S_m^p = F_m^p$ or $L_m^p, T_n^q = q^n, \quad e_{i,j} \in \{-1,0,1\}$, gives a so called signed-digit F-L p- representation.

- $S_m^p = F_m^p$ or $L_m^p, T_n^q = F_n^q$ or L_n^q, $e_{i,j} \in \{0,1\}$, gives a new, so called double F-L base p- representation.

These examples emphasize the multiplication representation systems of Fibonacci and Lucas numbers and their variations (i.e., non-weighted/ weighted F-L p-numbers), though they can be expanded to include Golden Ratio numbers and other systems. We note that it is also possible to fuse these representations using addition:

$$X = \sum_{i,j} e_{i,j} \{aS_m^{(p)} + bT_n^{(q)} + d\}, \quad p,q = 0,1,...,k-1, \qquad (14)$$

where a,b and d are constants. This linear combination of systems may lead to interesting representations such as:

$$X = \sum_k e_k \{a_k f_k + b_k l_k\} \text{ where } a_k \text{ and } b_k \text{ are constants: i.e., } a_k = \frac{1}{2}, b_k = \frac{1}{2} \text{ or } \frac{\sqrt{5}}{2} \qquad (15)$$

Definition: A number representation of an integer X is called a canonical or normal form if it is a standard way of presenting X. For example, the double base representation is a canonical form when $d = \sum_{n \geq 1} |e_n|$ is min.[15] Below we present an approach for determining the near canonical form of a certain two base representation.

Two Base Representations and the Greedy Algorithm: One way to find a certain two base representation of an n integer is to use the Greedy approach[11], which determines the best approximation of n, computes the difference, and reapplies the process. The general formula that describes this two base representation is

$$n = \sum_{i=1}^{l} s_i A_{a_i} B_{b_j}, \qquad (16)$$

where s_i is a set of predefined coefficients. Note this formula allows for the introduction of nontrivial coefficients in multiple base expansions. To illustrate the formula, let n= 841232. Setting $A_{a_i} = 3^{a_i} \, or \, F_{a_i}, B_{b_j} = 2^{b_j} \, or \, F_{bj}$, we can express n as a sum of double base terms or Fibonacci products using the Greedy Algorithm.

Double Base Expansion: $841232 = 2^7 3^8 + 1424 \rightarrow 1424 = 2^4 3^4 + 128 \rightarrow 128 = 2^7 3^0$
Fibonacci Expansion: $841232 = 832040 + 9192 \rightarrow 9192 = 8362 + 830 \rightarrow 830 = 754 + 76 \rightarrow 76 = 68 + 8$

Note that we call any system of expressing an integer n with mixed Fibonacci numbers in the form $\sum_{m,n} d_{m,n} F_m F_n$ the Double Fibonacci Base Number Representation System (referred to as FBNS or Fibonacci for short).

The Greedy Algorithm does not always produce a canonic representation. For example, consider the integer X=49 expressed in DFNS. The representation produced by the Greedy Algorithm is X=21*2+6*1+1*1. However, the canonic representation (only one such form exists for the integer 49) is X=13*3+5*2. Determining the canonical form of such a two base representation is often difficult, especially for very large integers. The Greedy Algorithm is thus used as a straightforward means of producing a near-to-canonic form. Integers represented using the Greedy Algorithm can have varying representation efficiencies based on the chosen two base representation system. That is, the minimal number of nonzero digits needed to represent each integer changes with respect to the chosen representation system of the form (16). In the following, we compare two representation systems generated from (16), namely the double base and the Fibonacci expansions, to see which system is more efficient in representing integers. We begin our comparison between the double base and Fibonacci expansions by analyzing the possible direct products or terms generated by each system.

Table of Double Base Products Less than 10,000

	1	3	9	27	81	243	729	2,187	6,561
1	1	3	9	27	81	243	729	2,187	6,561
2	2	6	18	54	162	486	1,458	4,374	13,122
4	4	12	36	108	324	972	2,916	8,748	26,244
8	8	24	72	216	648	1,944	5,832	17,496	52,488
16	16	48	144	432	1,296	3,888	11,664	34,992	104,976
32	32	96	288	864	2,592	7,776	23,328	69,984	209,952
64	64	192	576	1,728	5,184	15,552	46,656	139,968	419,904
128	128	384	1,152	3,456	10,368	31,104	93,312	279,936	839,808
256	256	768	2,304	6,912	20,736	62,208	186,624	559,872	1,679,616
512	512	1,536	4,608	13,824	41,472	124,416	373,248	1,119,744	3,359,232
1,024	1,024	3,072	9,216	27,648	82,944	248,832	746,496	2,239,488	6,718,464
2,048	2,048	6,144	18432	55,296	165,888	497,664	1,492,992	4,478,976	13,436,928
4,096	4,096	12288	36864	110,592	331,776	995,328	2,985,984	8,957,952	26,873,856
8,192	8,192	24576	73728	221,184	663,552	1,990,656	5,971,968	17,915,904	53,747,712

Table of Fibonacci Products Less than 10,000

	1	2	3	5	8	13	21	34	55	89
1	1	2	3	5	8	13	21	34	55	89
2	2	4	6	10	16	26	42	68	110	178
3	3	6	9	15	24	39	63	102	165	267
5	5	10	15	25	40	65	105	170	275	445
8	8	16	24	40	64	104	168	272	440	712
13	13	26	39	65	104	169	273	442	715	1,157
21	21	42	63	105	168	273	441	714	1,155	1,869
34	34	68	102	170	272	442	714	1,156	1,870	3,026
55	55	110	165	275	440	715	1,155	1,870	3,025	4,895
89	89	178	267	445	712	1,157	1,869	3,026	4,895	7,921
144	144	288	432	720	1,152	1,872	3,024	4,896	7,920	12,816
233	233	466	699	1,165	1,864	3,029	4,893	7,922	12815	20,737
377	377	754	1,131	1,885	3,016	4,901	7,917	12,818	20,735	33,553
610	610	1,220	1,830	3,050	4,880	7,930	12,810	20,740	33,550	54,290
987	987	1,974	2,961	4,935	7,896	12,831	20,727	33,558	54,285	87,843
1,597	1,597	3,194	4,791	7,985	12,776	20,761	33,537	54,298	87,835	142,133
2,584	2,584	5,168	7,752	12,920	20,672	33,592	54,264	87,856	142,120	229,976
4,181	4,181	8,362	12,543	20,905	33,448	54,353	87,801	142,154	229,955	372,109
6,765	6,765	13,530	20,295	33,825	54,120	87,945	142,065	230,010	372,075	602,085

Comparison of Double Base and Fibonacci Products Less than 10,000

Terms Below	Double Base	Fibonacci	Terms In Common
100	20	26	10
1000	40	58	13
10,000	67	101	14

These tables provide all double base and Fibonacci products less than 10,000. Note that terms <100, >100 and <1000, >1000 and <10,000 are represented with green, yellow, and orange cells, respectively. From the data tables, we can see that the Fibonacci and double base expansions are both very sparse. Still, the Fibonacci system seems to represent many numbers with fewer terms. We can attribute this to the fact that the Fibonacci system increases at a slower pace (though still quite quickly) than the double base system. As a result, more integers are represented as direct products of Fibonacci numbers than of binary and ternary numbers.

For example, we can see that there are 101 different Fibonacci products less than 10,000 (26 below 100, 32 below 1000 and above 100, and 43 below 10,000 and above 1000). In comparison, there are only 67 double base products less than 10,000 (20 below 100, 20 below 1000 and above 100, and 27 below 10,000 and above 1000). An interesting observation here is that as we allow the number limit to increase (say from below 1000 to below 10000), the number of Fibonacci products increases more than the number of double base products (double base: 20→ 27, Fibonacci: 32→43).

In other words, the difference in the number of Fibonacci and double base products increases as we allow our number range to increase (from 12=32-20 to 16=43-27). Generally, given its larger representation of numbers, which becomes more prominent as we allow our number range to increase, the Fibonacci product sequence is more efficient in representing numbers while maintaining a high degree of sparseness. We test this idea on various integers.

The first table below provides near canonical Fibonacci and double base expansions of integers ≤ 100 calculated using the Greedy Algorithm. In nearly half of the integers considered one method of expansion fares better than the other. That is, one system uses fewer terms to represent the respective integer. In 30 cases (represented by green cells), the Fibonacci system is more efficient in representing the given integer. In contrast, the double base system is more efficient in only 19 cases (represented by red cells). For the remaining cases (represented by non-shaded cells), both representation systems use the same number of terms to express the given integer. We can continue testing this idea.

From the table second table below, we see that the Fibonacci system is more efficient 37 times (represented by green cells) while the double base is more efficient only 21 times (represented by red cells). This data further affirms the idea that Fibonacci systems are more often than not more efficient in representing given integers. That is, the Fibonacci systems represent many more numbers with fewer terms than double base systems. As the number of Fibonacci products increases more than the number of double base products, we expect the Fibonacci system to generally become even more efficient as our given integer value increases (of course, there are exceptions to this expectation: consider, for instance, the above example X=841232 discussed above).

Table of Near Canonical Fibonacci and Double Base Expansions of Integers ≤ 100

	1	2	3	4	5	6	7	8	9	10
Double Base	1	2	3	4	4+1	6	6+1	8	9	9+1
Fibonacci	1	2	3	4	5	6	6+1	8	9	10
	11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Double Base	9+2	12	12+1	12+2	12+3	16	16+1	18	18+1	18+2
Fibonacci	10+1	10+2	13	13+1	15	16	16+1	16+2	16+3	16+4
	21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Double Base	18+3	18+4	18+4+1	24	24+1	24+2	27	27+1	27+2	27+3
Fibonacci	21	21+1	21+2	24	25	26	26+1	26+2	26+3	26+4
	31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
Double Base	27+4	32	32+1	32+2	32+3	36	36+1	36+2	36+3	36+4
Fibonacci	26+5	26+6	26+6+1	34	34+1	34+2	34+3	34+4	39	40
	41	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**
Double Base	36+4+1	36+6	36+6+1	36+8	36+9	36+9+1	36+9+2	48	48+1	48+2
Fibonacci	39+2	42	42+1	42+2	42+3	42+4	42+5	42+6	42+6+1	42+8
	51	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**
Double Base	48+3	48+4	48+4+1	54	54+1	54+2	54+3	54+4	54+4+1	54+6
Fibonacci	42+9	42+10	42+10+1	42+10+2	55	55+1	55+2	55+3	55+4	55+5
	61	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**
Double Base	54+6+1	54+8	54+9	64	64+1	64+2	64+3	64+4	64+4+1	64+6
Fibonacci	56+6	56+6+1	63	64	65	65+1	65+2	68	68+1	68+2
	71	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**	**80**
Double Base	64+6+1	72	72+1	72+2	72+3	72+4	72+4+1	72+6	72+6+1	72+8
Fibonacci	68+3	68+4	68+5	68+6	68+6+1	68+8	68+9	68+10	68+10+1	68+10+2
	81	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**
Double Base	81	81+1	81+2	81+3	81+4	81+4+1	81+6	81+6+1	81+8	81+9
Fibonacci	68+13	68+13+1	68+15	68+16	68+16+1	68+16+2	68+16+3	68+16+4	89	89+1
	91	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**
Double Base	81+9+1	81+9+2	81+12	81+12+1	81+12+2	96	96+1	96+2	96+3	96+4
Fibonacci	89+2	89+3	89+4	89+5	89+6	89+6+1	89+8	89+9	89+10	89+10+1

.

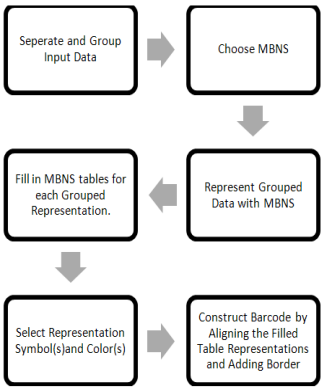
Table of Near Canonical Fibonacci and Double Base Expansions of Integers 101-200

	101	102	103	104	105	106	107	108	109	110
Double Base	96+4+1	96+6	96+6+1	96+8	96+9	96+9+1	96+9+2	108	108+1	108+2
Fibonacci	89+10+2	102	102+1	104	105	105+1	105+2	105+3	105+4	110
	111	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**	**120**
Double Base	108+3	108+4	108+4+1	108+6	108+6+1	108+8	108+9	108+9+1	108+9+2	108+12
Fibonacci	110+1	110+2	110+3	110+4	110+5	110+6	110+6+1	110+8	110+9	110+10
	121	**122**	**123**	**124**	**125**	**126**	**127**	**128**	**129**	**130**
Double Base	108+12+1	108+12+2	108+12+3	108+16	108+16+1	108+18	108+18+1	128	128+1	128+2
Fibonacci	110+10+1	110+10+2	110+13	110+13+1	110+15	110+16	110+16+1	110+16+2	110+16+3	110+16+4
	131	**132**	**133**	**134**	**135**	**136**	**137**	**138**	**139**	**140**
Double Base	128+3	128+4	128+4+1	128+6	128+6+1	128+8	128+9	128+9+1	128+9+2	128+12
Fibonacci	110+21	110+21+1	110+21+2	110+24	110+25	110+26	110+26+1	110+26+2	110+26+3	110+26+4
	141	**142**	**143**	**144**	**145**	**146**	**147**	**148**	**149**	**150**
Double Base	128+12+1	128+12+2	128+12+3	144	144+1	144+2	144+3	144+4	144+4+1	144+6
Fibonacci	110+26+5	110+26+6	110+26+6+1	144	144+1	144+2	144+3	144+4	144+5	144+6
	151	**152**	**153**	**154**	**155**	**156**	**157**	**158**	**159**	**160**
Double Base	144+6+1	144+8	144+9	144+9+1	144+9+2	144+12	144+12+1	144+12+2	144+12+3	144+16
Fibonacci	144+6+1	144+8	144+9	144+10	144+10+1	144+10+2	144+13	144+13+1	144+15	144+16
	161	**162**	**163**	**164**	**165**	**166**	**167**	**168**	**169**	**170**
Double Base	144+16+1	162	162+1	162+2	162+3	162+4	162+4+1	162+6	162+6+1	162+8
Fibonacci	144+16+1	144+16+2	144+16+3	144+16+4	165	165+1	165+2	168	169	170
	171	**172**	**173**	**174**	**175**	**176**	**177**	**178**	**179**	**180**
Double Base	162+9	162+9+1	162+9+2	162+12	162+12+1	162+12+2	162+12+3	162+16	162+16+1	162+18
Fibonacci	170+1	170+2	170+3	170+4	170+5	170+6	170+6+1	178	178+1	178+2
	181	**182**	**183**	**184**	**185**	**186**	**187**	**188**	**189**	**190**
Double Base	162+18+1	162+18+2	162+18+3	162+18+4	162+18+4+1	162+24	162+24+1	162+24+2	162+27	162+27+1
Fibonacci	178+3	178+4	178+5	178+6	178+6+1	178+8	178+9	178+10	178+10+1	178+10+2
	191	**192**	**193**	**194**	**195**	**196**	**197**	**198**	**199**	**200**
Double Base	162+27+2	192	192+1	192+2	192+3	192+4	192+4+1	192+6	192+6+1	192+8
Fibonacci	178+13	178+13+1	178+15	178+16	178+16+1	178+16+2	178+16+3	178+16+4	178+21	178+21+1

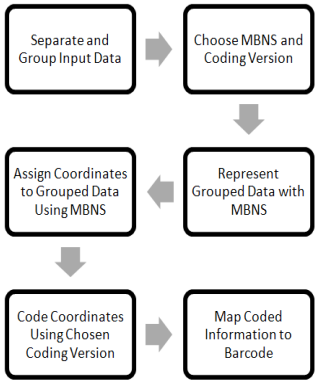
4. APPLICATIONS

We will now apply MBNS representations such as FBNS to construct more efficient and error resistant barcode systems. We consider 2D barcodes as they expand traditional 1D barcode information capacities by adding the vertical array of bars and spaces. By examination of mobile phone cameras and the normal use cases, the minimum requirements of the 2D codes for camera phones are identified[4]. They are a) Matrix codes are preferred to Stacked codes. b) Code size should grow proportionally to the data and have no sudden "jumps." c) Code should be easily detected and read regardless of its size and data amount. d) Code should have read flexibility so that it can be read under any angle. 2D code should also be efficient and widely supported. For instance, although 2D barcodes can be of any shape, most are made up of squares and rectangles due to pixel efficiency. Moreover, 2D barcodes are often not found in color and grayscale as black and white barcodes are more widely supported and allow for faxing and photocopying without losing readability. Taking these criteria into account, we propose representing the input data with MBNS representations such as DBNS and FBNS. The procedures for doing so are outlined in the flow chart below:

Algorithm 1:



Algorithm 2:



Note that for enhanced security, both algorithms may also use encryption techniques. As MBNS representations are sparse, they are efficient in representing given data and can lead to high performance barcodes. In general, these systems are also well equipped for error correction. In the following, we test our algorithms against the traditional base 2 representation used in QR code[3]. Though our algorithms are explained through a numeric example, they can be extended to different types (numeric, alphabetic, alpha-numeric, etc.) and languages of information.

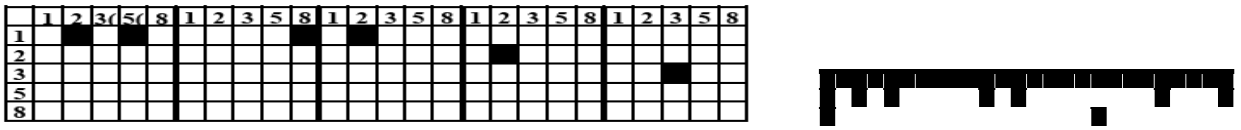
Algorithm 1 (Illustrative Example): Using the Double Base Number System to Represent Input Data "78249"

1. Separate the data "78249" into one bit groups.
2. Choose a representation system: DBNS or FBNS.
3. Determine each groups DBNS representation, for instance by using the Greedy Algorithm.
4. Fill in MNBS tables for each group representation and align them adjacently to one another. Though we can select from various representation symbols and colors, we here construct our barcode from black and white squares for pixel efficiency. We also add a boarder to our barcode by an outlining row and column. This way we can easily preserve the table format that we used in our generation.

DBNS Table and Barcode

	$1\ (3^0)$	$3\ (3^1)$	$9\ (3^2)$	$1\ (3^0)$	$3\ (3^1)$	$9\ (3^2)$	$1\ (3^0)$	$3\ (3^1)$	$9\ (3^2)$	$1\ (3^0)$	$3\ (3^1)$	$9\ (3^2)$	$1\ (3^0)$	$3\ (3^1)$	$9\ (3^2)$
$1\ (2^0)$	■														■
$2\ (2^1)$		■					■								
$4\ (2^2)$										■					
$8\ (2^3)$				■											

FBNS Table and Barcode*



*Note that because of repetition, we can construct modified FBNS tables without losing representation possibilities. In this case, we can ignore the last two rows (or columns). We can also represent the number 9 as 8*1+ 1*1 instead of as 3*3, and get rid of one additional column (or row). The FBNS barcode above is generated using this idea.

Algorithm 2: We now provide several versions of Algorithm 2 and apply them to coding the data "78249." Note that due to redundancy in systems such as FBNS, we are often not limited to shading in a particular cell and have the possibility of choosing a cell we find more conducive to easy reading (i.e., a cell that is not adjacent to any other filled cells).

Coordinate	Code
0	00
1	01
2	10
3	11

Version A

Coordinate	Code
0	0011
1	0101
2	1001
3	0110
4	1010

Version B

Number of Coordinates	Code
1	01
2	10
3	11

Version C

Coordinate	Code
0	0011
1	0101
2	1001
3	0110
4	1010
5	1100

Number of Coordinates	Code
1	00
2	01
3	10
4	11

Version D

Coordinate	Code
0	0000
1	0011
2	0101
3	1001
4	0110
5	1010
6	1100
7	1111

Code Version	*Double Base*	*Fibonacci*	*Double Base* **Barcodes**	*Double Base* **Barcodes**	*Fibonacci* **Barcodes**	*Fibonacci* **Barcodes**
A	10100011000 100100000					
B	10011001001 10110000000 00000000000 0000000	1001101000 1110100000 0000000000 0000000000				
C	10100101100 10100000000 0000000000					
D	11110000000 01100000000 00000000000 00000000000 00000000	1111100111 0010000000 0000000000 0000000				

As the charts above illustrate, we can vary many of elements of our algorithms including the number of elements in a group, the representation system, the code for representing coordinates, the matrix holding the code representation, and the barcode shape, color, and boarder. We can also vary the number of representation systems to construct higher dimensional barcodes. For instance, we can us three base representations (instead of the two base used here) to construct 3-D barcodes.

Error Correction and Code Versions*:* The code versions are generated using Hadamard matrices[16], which can be used to define some error correction codes. Note that code versions may also be generated to use other error

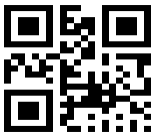
correction codes such as Reed-Solomon code. Though Version A may be the most compact, it may not be used for all MBNS systems and does not provide the error correction capabilities of the others. Hadamard code can be used to correct $n/4 - 1$ errors and detect $n/4$ errors in an n-bit encoded block. The table below illustrates some of the various levels of error correction using versions of Algorithm 2. Though code that can reach a data correction level of 15% is regarded as satisfactory[17], we can reach a level of up to 25%.

Levels of Error Correction Using Algorithm 2

	# Bits/ Coordinate	Error Detection Capability		Error Correction Capability	
Version		Bits	%	Bits	%
A	2	-	-	-	
B,C,D	4	1	25	-	
E	8	2	25	1	12.5
F	16	4	25	3	18.75
…	…	…	…	…	…

Comparison to QR Code*:* We now test our method against that used in QR Code. In order to make this comparison, a brief overview of how QR code is generated is provided below. Input data in QR Code can be represented in several modes, as illustrated in the table below. Each mode has a corresponding four-bit-long binary representation as well as a separate bit count for a group of input characters. For instance, the binary representation and reserved bit count for mode numeric is 0001 and 10, respectively; for alphanumeric mode, 0010 and 9, respectively. The binary representation corresponding to the input data is placed at the beginning of the encoded QR code. For instance, if we were to encode the zip code "78249", we would begin the code with the bits 0001 and reserve 10 bits for every group of numeric characters in the input data. Below we finish encoding the zip code "78249" in order to specifically illustrate how QR encoding works in numeric mode.

In order to represent 78249 in numeric mode, we group the input data with characters or bits of 3 or less. The data "78249" is separated into one 3 bit and one 2 bit group: "782" and "49". Each full 3 bit group is reserved a 10 bit binary representation. However, if the group contains fewer characters, say 2 or 1 bits, then a 7 or 4 bit binary representation is used respectively. Encoding "782" and "49" in 10 bit and 7 bit binary representations, respectively, we get: 1100001110 0110001. Adding on the binary representation corresponding to the QR numeric mode gives: 0001 1100001110 0110001. To finish encoding the examples, we take the representation arrange it into 8 bit groups. 0's are added to groups at the end of these representations that have less than 8 characters: "78249" $\rightarrow$00011100 00111001 10001000. This is done to fit the size of the QR code, which varies according to which version is used. QR code has versions from 1 to 40, where version 1 (used here) is a 21x21 matrix and each following version increases in length and width by 4 cells so that version 40 is a 177x177 matrix. It should be noted that QR code is utilized up to version 10 for camera phones as a greater density exceeds the capabilities of a camera as an imager and thus cannot be successfully decoded (Kato, Tan). This final code can be mapped to a barcode to form:



QRcode has four levels of error correction: L, M, Q, and H. About 7%, 15%, 25%, and 30% or less of errors can be corrected with each respective level. It is implemented using Reed-Solomon Code, which requires twice the amount of codewords to be corrected.

In the presented example, the total codewords that need to be corrected are 24, requiring 48 additional error correcting codewords. In comparison, Algorithm 1 and 2 use 50 and 20 codewords, respectively, to encode the same data without error correction. At this stage, both algorithms offer exceptionally easy reads and security features. Error correction features can be added. For instance, we can achieve a similar error correction level using Version E in Algorithm 2 with only 70 codewords. Note that the custom design capability of both algorithms, including variations in shape, size, color, and coding representation, allows the symbol to provide additional security. A more

general comparison of how well QR code, Algorithm 1, and Algorithm 2 satisfy the criteria that optimize 2D barcodes for mobile phones[17] is provided below.

Comparison of QR Code, Algorithm 1, and Algorithm 2

	Data Capacity	Error Correction	Scanability			Design			Security	Additional Features
			Omni-Directional	Low (VGA) Resolution	High Speed	Divisible	Scalable	Custom		
QR Code	+	+	+	+	+	+	-	-	-	Small printout size
Al- 1	+	-	+	+	+	+	+	+	+	Easy to read, especially at a distance. Legible under varying lighting conditions.
Al- 2	+	+	+	+	+	+	+	+	+	Small printout size. Provides additional error correction and encryption capabilities

5. CONCLUSION

In this paper, we introduced a concept of fusing parametric number representation systems. We determined that these systems provide sparse, efficient, and potentially secure means of representing data. Using the features of these fused systems, we created two algorithms that generate high data capacity barcodes. These barcodes are easy to read, scalable in size, potentially divisible, and omni-directional. Because the amount and type of information within the symbol is user selectable, the user can determine the density of the barcode and thus solve resolution problem of camera phones. Moreover, by having the option of selecting various representation systems, the user is offered a secure means of representing data. One can use the algorithms' custom design capabilities, in which symbol characteristics such as shape, size, and color can be varied, to provide additional security and carry more information. The generated barcodes also provide high percentage error correction and detection capabilities using the Hadamard or other error correction codes. By comparison, we found that many popular 2D barcodes, namely QR code, do not exhibit all these features. Finally, the proposed algorithms provide a viable alternative to current 2D barcodes and show a new direction in developing secure, reliable, and high capacity color and gray level barcodes.

REFERENCES

[1] Youssef, S.M. and Salem, R.M., "Automated barcode recognition for smart identification and inspection automation", Expert systems with applications: An international journal 33(4), 968-977 (2007).

[2] Meng, J. and Yang, Y., "Application of Mobile 2D Barcode in China", 4th International Conference on Wireless Communications, Networking and Mobile Computing, 1-4 (2008)

[3] Ohbuchi, E. and Hanaizumi, H. and Hock, L., "Barcode Readers using the Camera Device in Mobile Phones", Proceedings of the 2004 Intl. Conf. on Cyberworlds, 260-265 (2004)

[4] Wang, H. and Zou, Y., "Camera Readable 2D Bar Codes Design and Decoding for Mobile Phones", ICIP, 469-472 (2006)

[5] Rohs, M., "Real-World Interaction with Camera- Phones", 2nd International Symposium on Ubiquitous Computing Systems, 74-89 (2004)

[6] Ningzhong, L. and Han, S., “Design and Analysis of the Three-Dimensional Bar Code”, International Conference on Computer Science and Software Engineering 3, 669- 672 (2008)

[7] Stakhov, A. P., [Mathematics of Harmony: From Euclid to Contemporary Mathematics and Computer Science], World Scientific, (2008)

[8] Reitwiesner, G., “Binary arithmetic”, Advances in computers 1, 231–308 (1960)

[9] Joye, M. and Yen, S.M., “Optimal left-to-right binary signed digit recoding”, IEEE Trans. on Computers 49(7), 740-748 (2000)

[10] Parhami, B., “Generalized Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations,” IEEE Transactions on Computers 39(1), 89-98 (1990)

[11] Dimitrov, V.S. and Jullie, G. A., “A New Number Representation with Applications**”,** IEEE Circuits and Systems Magazine 3(2), 6-23 (2003)

[12] Zazkis, R. and Campbell, S., “Prime decomposition: Understanding uniqueness”, Journal of Math Behavior 15(2), 207-218 (1996)

[13] Clark, W. E. and Liang, J., “On arithmetic weight for a general radix representation of integers,” IEEE Trans. Inform. Theory 19, 823-826 (1973)

[14] Morain, F. and Olivos, J., “Speeding up the computations on an elliptic curve using addition-subtraction chains”, Theoretical Informatics and Applications 24, 531–543(1990)

[15] Heuberger, C., **“**Minimal Expansions in Redundant Number Systems: Fibonacci Bases and Greedy Algorithms”, Periodica Mathematica Hungarica 49(2), 65–89 (2004)

[16] Horadam, K.J., [Hadamard matrices and their applications], Princeton University Press, (2007)

[17] Kato, H. and Tan, K.T., “2D barcodes for Mobile Phones”, Proceedings of 2nd International Conference on Mobile Technology, Applications and. Systems, 1-8(2005)