International Journal of Computer Science and Applications

Vol. 8 No. 1, pp. 136 - 155, 2011

2D COLOR BARCODES FOR MOBILE PHONES *

M. QUERINI, A. GRILLO, A. LENTINI and G.F. ITALIANO

Department of Computer Science, Systems and Production
University of Rome "Tor Vergata"
via del Politecnico 1, 00133 Rome, Italy
marco.querini@uniroma2.it
grillo;lentini;italiano@disp.uniroma2.it

We propose a new high capacity color barcode, named HCC2D (High Capacity Colored 2-Dimensional), which use colors to increase the barcode data density. The introduction and recognition of colored modules poses some new and non-trivial computer vision challenges, such as handling the color distortions introduced by the hardware equipment that realizes the Print&Scan process. We developed a prototype for generating and reading the HCC2D code format, both on desktops (Linux and Windows platforms) and on mobile phones (Android platforms). We tested this prototype in many experiments considering different operating scenarios and data densities, and compared it to known 2-dimensional barcodes.

Keywords: 2D barcodes; Color barcodes; Print&Scan Process; Mobile Devices; Mobile Applications.

1. Introduction

Barcodes are optical machine-readable representations of data, capable of storing digital information about the physical object to which they are attached. Due to their reading speed, accuracy, and functional characteristics, barcodes have become ubiquitous in many applications, including their usage in department stores and retail chains to price goods, to track items and to identify customers through membership cards; in tracking item shipment and movement, such as express mail, rental cars, airline luggage; in patient identification in hospitals; in document management systems; in ticketing for sports events, cinemas, theaters and transportation.

Traditional barcodes, referred to as one-dimensional (1D) barcodes, represent data by varying the widths and spacings of parallel lines. The amount of digital information stored in 1D barcodes is limited and could be simply increased by increasing the number of barcode digits or by laying out multiple barcodes. This approach has many negative effects, however, such as enlarged barcode areas,

*Work partially supported by the EU under Contract no. FP7-SME-2010-1-262448 (Project SIGNED). A preliminary version of this paper was presented at the International Multiconference on Computer Science and Information Technology [Grillo *et al.* (2010)].

Fig. 1. Dimension for storing data in (a) 2D and (b) 1D codes.

Fig. 2. Evolution of barcodes: (a) multiple barcode layout, (b) stacked barcode layout, and (c) matrix barcode layout.

more complex reading operations, and increased printing costs. For this reason, the barcode technology has been deploying geometric patterns (such as squares, dots, triangles, hexagons) in two dimensions: such barcodes are referred to as bidimensional (2D) codes. Note that 2D codes increase the data space available by storing information in two dimensions, whereas 1D codes contains data in one dimension only. Figure 1 shows examples of 1D and 2D barcodes.

Available 2D codes span from repeating a single 1D barcode over multiple rows to exploiting bidimensional shapes in order to represent data. Figure 2 illustrates the evolution of 2D barcode technology. In particular, Figure 2 (a) shows a multiple barcode layout: the main disadvantage related to this simple 2D layout is the need of multiple scans in order to get all the information contained in the barcode. Figure 2 (b) illustrates a stacked barcode layout: in this case one single scan is enough to obtain the stored information but the scanning equipment must be carefully aligned with the barcode orientation. Finally, in Figure 2 (c) a matrix barcode layout is presented: this layout enables to acquire information with one single scan and does not require the accurate alignment of the scanning equipment.

There are more than 20 types of conventional 2D codes. Figure 3 illustrates some examples of 2D codes; one of the main differences among those barcodes lies in the amount of data which can be stored in a single barcode. For example, the

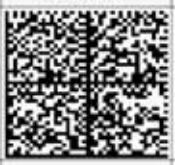
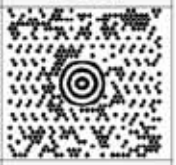
	QR Code	PDF417	DataMatrix	Maxi Code
				
Developer (country)	DENSO (Japan)	Symbol Technologies (USA)	RVSI Acuity CiMatrix (USA)	UPS (USA)
Code Type	Matrix	Stacked Bar Code	Matrix	Matrix
Numeric Data	7,089	2,710	3,116	138
Alphanumeric Data	4,296	1,850	2,355	93
Binary Data	2,953	1,018	1,556	
Kanji Data	1,817	554	778	
Main features	Large capacity, Small printout size, High speed scan	Large capacity	Small printout size	High speed scan
Standardization	AIM International, JIS, ISO	AIM International, ISO	AIM International, ISO	AIM International, ISO

Fig. 3. Characteristics of some types of 2D codes.

Quick Response (QR) code[a] is able to store more than 7,000 decimal digits, while Maxi code is able to store only 138 decimal digits.

Due to the plethora of applications for barcodes, there is an emerging need for barcodes capable of storing even more information and more character types in smaller printing space. Furthermore, the wide diffusion of smartphones equipped with low-end cameras introduces new challenges for barcodes and makes it appealing to apply this technology to mobile applications. Just to mention only one application, the International Air Transportation Association (IATA) recently announced a global standard for global mobile phone check-in using 2D barcodes [IATA (2007)]; this will allow airlines to send boarding passes as 2D barcodes to the customers' mobile devices, thus eliminating completely the need for paper travel documents.

Both the increasing demand for higher density barcodes and the wide availabil-

[a]QR code is a 2D code developed by Denso Wave, a division of Denso Corporation at the time, and released in 1994 with the primary aim of being easily interpreted by scanner equipment. QR code is registered trademark of Denso Wave Incorporated in Japan and other countries.

ity of on-board cameras in mobile phones seem to motivate naturally the need for 2D color barcodes. However, the increased data density obtained with the usage of colors comes at an additional cost. Today, a Print&Scan process is commonly used for image reproduction and distribution, where images are converted between printed and digital formats. A rescanned image may look similar to the original, but it may have been distorted during the Print&Scan process. Indeed, reading color barcodes poses significant computer vision challenges [Parikh and Jancke (2008); Bulan *et al.* (2009); Siong *et al.* (2008)]. This is due to several factors, and we cite only few of them in the following. First, the color balance may be drastically different in different code readers. Second, the images containing codes may be taken by unexperienced users, and thus the location of the barcode in the image, its orientation, its slope, etc. can be mostly unconstrained. Furthermore, possible transformations in the prospective can distort the geometry of the barcode. Last but not least, the light conditions under which the images are taken can vary dramatically.

To the best of our knowledge, up to date only three color barcodes have been proposed in the literature: the colored DataGlyphs developed at Xerox Parc [Hecht (2001a); Hecht (2001b)], the High Capacity Color Barcode (HCCB) developed at Microsoft [Microsoft Research (2007); Parikh and Jancke (2008)] and the high capacity color barcode technique proposed in [Bulan *et al.* (2009)]. As already observed in [Bulan *et al.* (2009)], DataGlyphs fundamentally offer the same capacity as standard black and white barcodes. On the other hand, the method proposed in [Bulan *et al.* (2009)] is based on printing two colors at the same spatial location, and thus at most nearly doubles the capacity of black and white barcodes. The HCCB barcode of Microsoft encodes data as triangles and uses different colors, where the color chosen for each triangle is data-dependent. While HCCB is considered to be one of the leading barcodes in data density, it seems to be less robust than other 2D codes in the scanning phase, since it lacks explicitly patterns to support the detection and alignment process. We mention that the HCCB barcode format has not been evolved for several years, and its main application has been Microsoft Tag [Jancke (2011)], which is essentially a machine readable web link.

In this paper, we propose a new high capacity color barcode, named HCC2D (High Capacity Colored 2-Dimensional), which use colors to increase the barcode data density. The introduction and recognition of colored modules in HCC2D poses some new and non-trivial computer vision challenges, such as handling the color distortions introduced by the hardware equipment that realizes the Print&Scan process. The HCC2D codes presented in this paper are able to support different types and sizes of data input, and adapt smoothly the code dimension to the actual input size. In order to support all those scenarios in which the Print&Scan process imposes the usage of only two colors (i.e., black and white) HCC2D considers black and white barcodes as barcodes with exactly two colors. In particular, HCC2D has been designed so as to be fully compatible with the QR code, which is currently among the most widespread 2D barcodes (a QR code represents the simplest case

of our 2D colored code). The main advantage of HCC2D over QR is that HCC2D is able to store substantially more data than QR, while preserving the strong reliability and robustness properties of QR.

We developed a prototype for generating and reading the HCC2D code format, which is able to realize the entire Print&Scan process, and runs both on desktops (Linux and Windows platforms) and on mobile phones (Android platforms). We tested our prototype in many experiments considering different operating scenarios and data densities on low-end hardware, such as inkjet printers and inexpensive mobile phones. In our experiments, as a first step, we attempted to measure the data density and the computational overhead introduced by HCC2D. Our first experimental findings show that at 600 dpi HCC2D barcodes with 8 colors can achieve data densities of 15,048 bits per square inch, which is very close to the value of 16,000 bits per square inch reported for HCCB with 8 colors at 600 dpi [Microsoft Research (2007)]. As for performance issues, since HCC2D is fully compatible with QR codes, we measured the computational overhead introduced by HCC2D over QR codes during the entire reading process, which appears to be the most critical step for color barcodes in the Print&Scan process. Our experiments show that the increased data density of HCC2D seems to come at a reasonable price: indeed in all our tests, HCC2D codes with up to 16 colors and different error-correction levels are about only 150 msec slower than their corresponding QR codes, with a relative computational overhead between 3% and 38% with respect to QR codes. In summary, HCC2D codes obtain data densities close to HCCB (one of the leading barcodes in data density) and strong robustness similar to QR codes.

The remainder of this paper is organized as follows. We describe the QR codes in Section 2, and the HCCB codes in Section 3. In Section 4 we introduce our new HCC2D codes and discuss their main features. We further discuss the results of some experiments with our prototype for HCC2D codes in Section 5. We end with some concluding remarks in Section 6.

2. The QR code

QR codes as well as other black and white 2D codes store data using a graphical representation. The core of this representation is based on the arrangement of multiple simple geometric shapes over a fixed space. A generic 2D code is required to perform efficiently at least the following three functions:

- the position detection function is a critical function; elements that serve as position detection function give to the acquisition process the capability of identifying the presence of a 2D code in the acquired image;
- the alignment function is required to synchronize the Scan process on the correct position of a 2D code. This function exploits some alignment patterns placed by the Print process in a well known position. Hence, the Scan process focuses on retrieving these well known patterns in order to position correctly the 2D code.

- the data function is required to encode the input data in a specific graphical representation. Some additional goals may be reached by the data function; error correction and data masking are examples of these functions for strengthening the 2D code.

QR codes adopt an arrangement of black and white squares of different sizes for all the required functions. In particular, each module represents a single bit following a simple rule: black squares store 1 and white squares store 0. In the following, some of the features provided by QR codes will be discussed.

High Capacity Encoding of Data. While conventional 1D codes store up to 20 decimal digits, the QR code is able to store from several dozen to several hundred times more data. QR codes can handle a large variety of data, such as binary, numeric and alphabetic characters, Kanji, Kana and Hiragana (Japanese) symbols, and control codes. If the input is represented by decimal digits, one symbol can encode up to 7,089 decimal digits (see Figure 3).

Small Printout Size. Since QR codes store information both horizontally and vertically, they are capable of encoding the same amount of data in approximately one-tenth the space of a traditional 1D code.

Dirt, Damage and Distortion Resistant. QR codes have error correction capability. Data can be restored even if the symbol is partially dirty and damaged. A maximum of 30% of the codewords can be restored. A codeword is a unit that constructs the data area. In the case of a QR code, one codeword is equal to 8 bits. Thanks to their alignment function, QR codes are resistant to distorted acquisitions.

Readable from any direction in 360 degrees. QR codes are capable of 360 degrees (omni-directional) reading. This task is accomplished through position detection patterns located at the three corners of the symbol. These position detection patterns guarantee stable high-speed reading, circumventing the negative effects of background interference.

Structured Append Feature. If a single QR code is too large for the print space available, a splitting function may be applied to obtain smaller QR codes containing the same data. One data symbol can be divided into up to 16 codes, which can be printed in smaller areas.

Standardization Process. QR codes have become widely used for two reasons: QR code specifications are clearly defined and made public and the QR codes can be freely usable. QR code is open in the sense that the specification of QR code is disclosed and that the patent right owned by Denso Wave is not exercised.

More in detail, a QR code can be represented as shown in Figure 4; the structure is composed of some elements that perform the various functions. The available space in each symbol may serve as *Function Patterns* or as *Encoding Region*. The *Position Detection Patterns*, the *Alignment Patterns*, the *Timing Patterns*, and the *Separators for Position Detection Patterns* support the Scan process in detecting

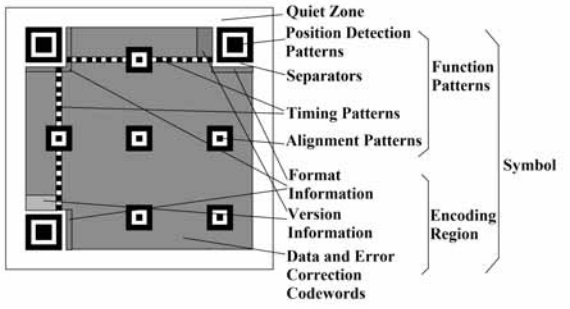


Fig. 4. Structure of a generic QR code.

the presence, the proper orientation and the correct slope of a QR code into an image. The *Format Information* describes the error correction level used in the code; it is possible to use four different error correction levels in the QR code: the lower level (i.e., Level L) is able to correct about 7% of the data, the Level M restores about 15% of the data, the Level Q is able to fix about 25% of the data, and the higher level (i.e., Level H) corrects approximately 30% of the data. The *Version Information* contains the real size of the code; it is possible to generate QR codes starting from Version 1 (i.e., 21x21 modules) to Version 40 (i.e., 177x177 modules). Finally the *Data and Error Correction Codewords* contains input and error correction data.

3. The HCCB code

We next describe briefly the main features of the HCCB code, a color barcode introduced by Microsoft [Microsoft Research (2007); Parikh and Jancke (2008)]. HCCB stands for High Capacity Color Barcode; it consists of rows of triangles of (up to eight) different colors, where consecutive rows are separated by a white line (see Figure 5). While the number of rows in a HCCB code may vary, the number of modules in each row is always a multiple of the number of rows. A module represents the basic entity for storing information in a 2D code. The HCCB code has a black boundary around it, further surrounded by a thick white band: these patterns are designed to act as visual landmarks in order to locate the barcode in an image. The black boundary at the bottom of HCCB is thicker than the boundaries on the other three sides: the bottom boundary acts as an orientation landmark, as barcodes may be at an arbitrary orientation in the image. The last triangles in the last row are always in a fixed order (2 triangles per color) and are used as a color palette during the scan, i.e., they are used to reconstruct colors during the detection phase (so that they are less prone to the chromatic distortion introduced during the printing phase). The detection process of an HCCB code works as follows: it starts from a point which is supposed to be at the interior of the code and proceeds on squares

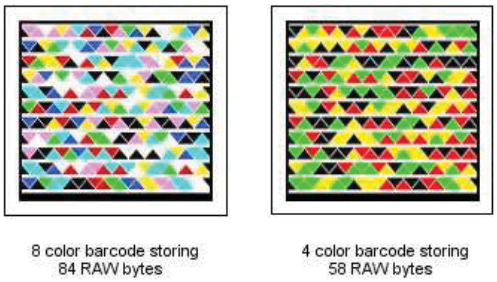


Fig. 5. An example of the Microsoft High Capacity Color Barcode (HCCB) (Viewed better in color).

of larger sizes until it recognizes the white border around the code; after the white border has been located, it starts the alignment process by looking for the thick bottom boundary.

One possible disadvantage of HCCB codes is related to their fragility in the detection and alignment process. First of all, the automatic recognition of an HCCB code can be problematic, since the position detection could start from any image contained inside a white border (which is not necessarily an HCCB code), thus giving rise to delayed failures. Furthermore, the availability of only one color palette makes the code more vulnerable to dirt, distortions and damages in the area containing the color palette. Finally, during either the print or the scan phase, because of distortions, rows can change their slope (and thus could not be perfectly aligned along a straight line): if the distortion is too big, this might result in failures to properly recognize the boundary.

4. High Capacity Colored 2-Dimensional Code

In this section, we describe our HCC2D code, which was designed with the main goal of increasing the data density while preserving the strong robustness and error correction properties of the QR codes. HCC2D defines a superset of the QR code set, and thus it is able to maintain fully compatibility with QR. In particular, HCC2D increases the data density by generating each module of the data area with a color selected from a color palette. Figure 6 illustrates samples of HCC2D with 4 and 8 colors.

We designed the HCC2D code preserving all the *Function Patterns*, the *Format Information* and the *Version Information* defined in the QR code. Maintaining the structure and the position of such critical information allows the HCC2D code to preserve compatibility with the QR code. Furthermore, the space required by all this information is small, so we did not reduce this space to increase the data density. Any modification to such information may led to failures in the recognition process. The most important changes are gathered in the

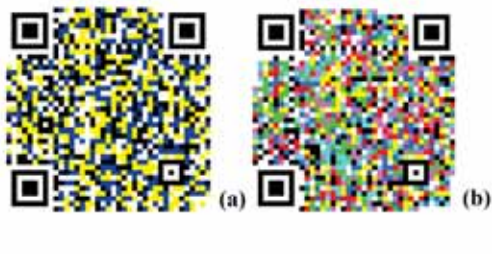


Fig. 6. Samples of the High Capacity Colored 2-Dimensional Code (HCC2D): (a) 4 colors, version 5H, and (b) 8 colors, version 4H (Viewed better in color).

Data and Error Correction Codewords area. The most noticeable difference with a QR code is that the modules may be of different colors; in a code with a palette composed by at least 4 colors each module is able to store more than one bit. Introducing colors in the data and error correction area requires to address some issues, which will be analyzed next.

4.1. *HCC2D Code Tables*

The HCC2D Code Tables contain some information such as the total codewords count, the symbol version, the error correction level, the Reed-Solomon block type, etc. The aim of these tables is to support users in selecting the best code once the size and kind of the input data and the desired error correction level are known. In order to define the table it is possible to refer to those published in the ISO/IEC 18004 document that contains the definition of the QR code. The Bits per Module (BpM) can be defined as the number of bits that a single module is able to store: $BpM = \log_2(\text{number of colors})$.

Version	**Number of modules**	**Function patterns modules**	**Format and Version info modules**	**Data modules**	**Data capacity (codewords)**	**Remainder bits**
1	21x21	202	31	208	26	0
2	25x25	235	31	359	44	7
3	29x29	243	31	567	70	7
4	33x33	251	31	807	100	7
5	37x37	259	31	1,079	134	7
6	41x41	267	31	1,383	172	7
7	45x45	390	67	1,568	196	0

Table 1. Data capacity for smaller version of QR codes.

In Table 1 the data capacity for the smaller versions of QR codes is detailed. Each codeword is one byte; when the total number of data modules available is not a multiple of 8, some bits are left unused: those are the *Remainder Bits*, which fill the empty positions after the final codeword. The more colors available, the more data can be stored into the code. Table 2 illustrates the variation in data capacity and remainder bits for the corresponding HCC2D codes.

Version	**Data Capacity (Codewords)**			**Remainder Bits**		
	4 colors (2 BpM)	8 colors (3 BpM)	16 colors (4 BpM)	4 colors (2 BpM)	8 colors (3 BpM)	16 colors (4 BpM)
1	52	78	104	0	0	0
2	88	132	176	14	21	28
3	140	210	280	14	21	28
4	200	300	400	14	21	28
5	268	402	536	14	21	28
6	344	516	688	14	21	28
7	392	588	784	0	0	0

Table 2. New values for data capacity for smaller version of HCC2D codes.

Version	**Error correction level**	**Effective data capacity (QR)**	**Effective Data Capacity**		
			4 colors 2 BpM (HCC2D)	**8 colors 3 BpM (HCC2D)**	**16 colors 4 BpM (HCC2D)**
1	L	19	38	57	76
1	M	16	32	48	64
1	Q	13	26	39	52
1	H	9	18	27	36
2	L	34	68	102	136
2	M	28	56	84	112
2	Q	22	44	66	88
2	H	16	32	48	64

Table 3. Error Correction Level for version 1 and version 2 of QR codes and HCC2D codes. Effective data capacities are shown in codewords.

Table 3 illustrates the effective data capacity for version 1 and version 2 of QR and HCC2D codes with a specific error correction level. Starting from these values and choosing the data type that will be encoded in the QR and HCC2D code yields the effective code capacity. We designed HCC2D codes so as to keep

the same error correcting capabilities as QR codes. Consider for instance QR and HCC2D codes with version 5 and error correction level H. The QR code has 134 codewords in total, out of which 88 are ECC (Error Correcting) codewords and 46 are data codewords. The QR code consists of 4 Reed Solomon blocks: the first two blocks contain 22 ECC codewords and 11 data codewords, while the last two blocks contain 22 ECC codewords and 12 data codewords. The Error Correction used (Reed Solomon) is able to correct up to 44 codewords, yielding a maximum ECC rate of $44/134 = 32.8\%$. The corresponding HCC2D code with 4 colors has 268 codewords in total, out of which 176 are ECC (Error Correcting) codewords and 92 are data codewords. The HCC2D code consists of 8 Reed Solomon blocks: the first four blocks contain 22 ECC codewords and 11 data codewords, while the last four blocks contain 22 ECC codewords and 12 data codewords. The Error Correction used is able to correct up to 88 codewords, yielding again a maximum ECC rate of $88/268 = 32.8\%$.

QR codes contain a *Mode Parameter*, which specifies the data type (numeric, alphanumeric, etc...) used in the symbol, and a *Character Count Indicator*, which stores the number of elements of that specific data type which are encoded in the symbol. Table 4 shows the number of bits available for the *Character Count Indicator* in QR codes. Namely, QR code versions from 1 (i.e., 21x21 modules) to 9 (i.e., 53x53 modules) in alphanumeric mode reserve 9 bits for the *Character Count Indicator*, and thus each symbol in alphanumeric mode can store at most 511 (i.e., $2^9 - 1$) characters.

Version	**Numeric Mode**	**Alphanumeric Mode**	**8-bit Byte Mode**	**Kanjii Mode**
1 to 9	10	9	8	8
10 to 26	12	11	16	10
27 to 40	14	13	16	12

Table 4. Number of bits reserved for the *Character Count Indicator* for the various QR code version.

HCC2D codes have higher data densities, and thus need higher values for the *Character Count Indicator*. In particular, when the color palette has only 2 colors (black and white), the number of bits for the *Character Count Indicator* is defined exactly as in QR codes (see Table 4). Otherwise, the number of bits reserved for the *Character Count Indicator* in a HCC2D code with at least 4 colors is defined as follows:

- for the *8-bit Byte Mode*, 16 bits are reserved, regardless of the HCC2D code version used;
- for the remaining modes, the number of bits reserved is $CCI_{QR} + \lceil BpM/2 \rceil$, where CCI_{QR} denotes the number of bits used by the QR *Character Count Indicator*.

4.2. *The Color Palette*

If the *Encoding Region* is composed of colored modules, the Scan process needs to know the complete color palette in order to decode the symbol. In the QR code during the Scan process only the brightness information is taken in account. A simple solution is to consider the color palette as an *a priori* shared knowledge between the Print and the Scan processes; in such a scenario handling the distortions introduced by the specific hardware (e.g., scanner, camera, ...) represents a critical issue. Hence, in the HCC2D code we have introduced an additional field, the color palette, to ensure that the Scan process is able to know how many and which colors are used in the scanned code. Encoding the color palette directly in the HCC2D code helps in reducing failures in the acquisition process due to the color distortions related to the different hardware used in the Print&Scan process.

In practical scenarios, it is important to define some quick failure criteria that avoid unnecessary computations if the Scan process delays a successful recognition. Forcing the start of a new Scan process instead of trying to recognize low quality images can reduce the delayed failure. Sorting the colors that compose the color palette according to the brightness value may represent a simple criterion to overcome the delayed failure problem. The Scan process starts recognizing only the color palette, if colors in the palette are not sorted as expected the process terminate with a quick failure, otherwise the Scan process can continue.

Furthermore, if the color palette is replicated around the bidimensional code, the quick failure rate can be further reduced: each color palette that does not respect the expected color sorting can be discarded. If a minimum number of color palettes is successfully recognized, the Scan process can build the color palette for decoding the code by computing the average of the valid color palettes. After analyzing the color palette in the HCC2D code, the Scan process is able to build a model for evaluating each module in the *Encoding Region*. In the QR code the problem of distinguishing between light and dark modules is addressed by handling only the brightness information; exploiting the *Quiet Zone* to calibrate the Scan process, an appropriate threshold is chosen. In HCC2D a more complex similarity function is needed for handling colored modules. Since colors in computer graphics are usually represented as vectors in multidimensional space (e.g., the Red Green Blue model, the Cyan Yellow Magenta Key model, the Hue Saturation Lightness model, etc.), we solve the identification problem by using Euclidean distances between such vectors. In particular, modules in the encoding region are considered as vectors according to the specific color model used by the scanning equipment; the color recognized is given by the color in the palette which minimizes the vector distance.

So far, we have introduced the need for a new kind of pattern, i.e., the *Color Palette Pattern*, which must be replicated in the symbol for reliability issues. We point out that this pattern is not inherited by the QR code design, but it is a new feature introduced to address the challenges posed by color recognition. HCC2D codes have such patterns in their boundaries (see Figure 7). Note that the *Color*

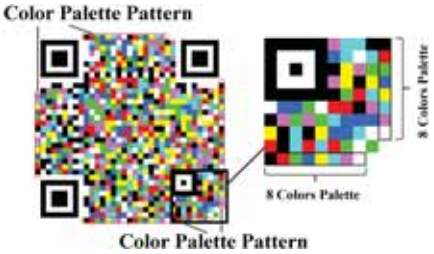


Fig. 7. The four *Color Palette Patterns* are pointed out in a HCC2D code (8 colors, version 4) sample. (Viewed better in color).

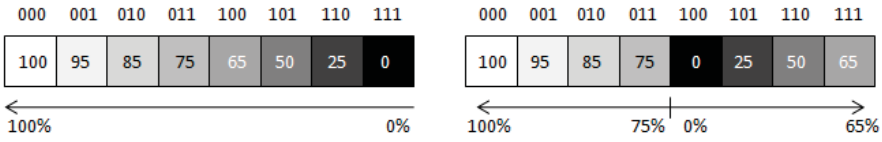
Palette Patterns are not too close to the three *Position Detection Patterns* areas and are far away from each other, thus ensuring that they are robust to local distortion. Furthermore, *Color Palette Patterns* take only 2 rows and 2 columns from a symbol consisting of between 21 and 177 rows and columns.

4.3. *Data Masking*

The main purpose of data masking in the QR code is to reach an appropriate balance between dark and light areas and to avoid that patterns similar to those used for *pattern Detection*, *Alignment* and *Timing* appear in the *Encoding Region*. The HCC2D needs a similar function for preserving the Scan process in recognizing the *Function Patterns* and increasing the code robustness. The QR code defines eight different masks; each mask is generated according to a simple rule that mimics the *Function Patterns*. The Print process has to select the mask that obtains the best score according to a scoring rule that is based on some bitwise XOR operations. Switching the brightness of some modules (i.e., light modules become dark modules and vice versa) is the result of data masking.

Since HCC2D increases the number of bits per module (BpM), we need to define new bitwise operations that are able to handle more than one bit per module. As far as the mask selection step is concerned, the HCC2D code is considered as a binary matrix composed simply of dark (i.e., the darkest colors) and light (i.e., the lightest colors) modules; the score for each mask is computed as in the QR code without taking in account the chromatic information of modules which is disjoint by the brightness information. Once the best mask is identified, each bit of the mask has to be used for switching the color of the module in the *Encoding Region*. For each module a bitwise operation between the bit in the mask and each bit that is stored in the module has to be performed.

To preserve the balance between dark and light areas, we need to properly organize the palette. The colors are sorted in descending order according to the brightness information, the lighter colors first and the darker colors last; the color palette is then splitted in two halves and the second half is sorted in reverse order.



start	switched	delta
000	111	100%
001	110	70%
010	101	30%
011	100	10%
100	011	10%
101	010	30%
110	001	70%
111	000	100%

start	switched	delta
000	111	35%
001	110	45%
010	101	60%
011	100	75%
100	011	75%
101	010	60%
110	001	45%
111	000	35%

Fig. 8. Color palette organization for data masking.

As shown in Figure 8, this simple reorganization of colors in the palette results in increasing the minimum brightness distance between switched colors from 10% up to 35%.

4.4. *Aspects related to Mobile Phones*

In this section we describe some features of HCC2D codes which are specifically related to their usage with mobile phones. We start by noticing that having color palettes in the boundaries of the symbol, as described in Section 4.2, helps to maintain the compatibility with simple QR codes. However, situations where the chromatic distortions in the internal areas and the boundaries of the symbol are different can pose some problematic issues during the recognition step (see Figure 9). Note that this might be a frequent case with flash-equipped camera phones, where the light source (flash) cannot be placed far away from the printed code. Since in these cases the projected light on the printed surface, and therefore the perceived color, degrades more or less gracefully, we tackle this issue by adjusting adaptively the color thresholds while we walk across the *Encoding Region* areas during the recognition step. This could be done by computing a moving weighted average between the perceived color palette and the perceived sample values in the *Encoding Region*. We notice that this method is more successful if we start by sampling modules close to the *Color Palette Patterns* (e.g, from one of the corners).

Another important challenge for mobile phones is related to video compression.

Fig. 9. A typical barcode reading challenge in a dark, barely lighted environment. With flash-equipped camera phones, camera and light are forced to be at the same distance from the printed surface (Viewed better in color).

Decoding color barcodes must meet the same challenges as black and white barcodes, such as geometric distortions, luminance balancing, variable light conditions and barcode localization. However, color barcodes introduce special challenges on their own. Two components of a video signal are *luma*, i.e., the brightness of an image (the achromatic portion), and *chroma*, i.e., the color information. In image compression, an image is stored in a YCbCr color space, where Y is related to the brightness and Cb and Cr are the color components. Since for the human eye the chroma signal is perceptually less important than the luma signal, digital video signals are typically compressed by sampling the color components (Cb, Cr) at a lower rate than the brightness (Y). This technique is referred to as *chroma subsampling*.

It is common to represent the ratios of information stored in these different channels of the YCbCr space as Y:Cb:Cr (describing how often Cb and Cr are sampled relatively to Y). For instance:

- 4:4:4 means no subsampling of the chroma channels.
- 4:2:2 means 2:1 horizontal subsampling, with no vertical subsampling. (Every scan line contains four Y samples for every two Cb or Cr samples).
- 4:2:0 means 2:1 horizontal subsampling, with 2:1 vertical subsampling.
- 4:1:1 means 4:1 horizontal subsampling, with no vertical subsampling. (Every scan line contains four Y samples for every Cb or Cr sample).

In other words, a 4:2:0 sampling pattern (which is a common camera preview format in many mobile phones) implies that if the brightness (Y) is sampled at 13.5 MHz, i.e., 13.5 million times each second, the chrominance is sampled at one-fourth the rate of the luminance, i.e., 3.375 Mhz for Cb and 3.375 Mhz for Cr. Figure 10 compares such a widespread format for video data compression, i.e., YCbCr 4:2:0, with the full rate sampling format, that is, YCbCr 4:4:4.

Although chroma subsampling does not affect the detection of black and white barcodes, it can be a major issue while reading color barcodes with mobile phones. To address this problem, we use a Euclidean distance, weighted so that it is more sensible to the luminance component than to the (subsampled) chromatic compo-

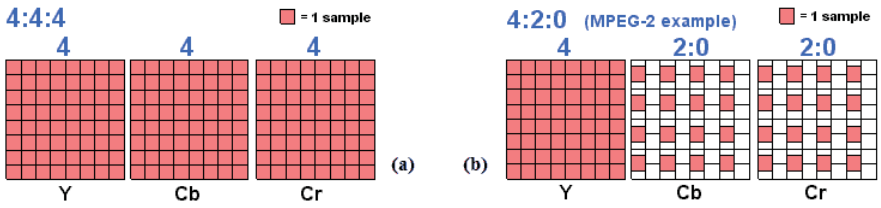


Fig. 10. Comparison between (a) YCbCr 4:4:4, according to which Cb and Cr are sampled at the same full rate as the luma and (b) YCbCr 4:2:0, the Android Camera Preview Default Format, characterized by subsampling chromatic informations (1 color sample per 4 luma samples).

nent. For performance issues, it is thus preferable to use colors with different values of brightness. Note that this can be naturally combined with a proper balance of dark and light areas, as discussed in Section 4.3.

5. Experimental Results

We developed a prototype that implements the Print&Scan process for HCC2D codes, working on both desktops and mobile phones. In particular, we implemented two different applications, the encoder and the decoder, for generating and acquiring HCC2D codes. The HCC2D code encoder was realized with the help of *libqrencode* [Fukuchi (2011)], a `C` library for encoding data in QR code symbols, while the decoder was built with the help of *zxing* [Google (2011)], an opensource Java project for improving the processing of 1D/2D barcodes. In our implementation, the decoder is able to recognize QR codes as well; some of the operations executed in the acquisition process are preserved: recognizing the position detection pattern, recognizing and exploiting the alignment patterns, and reading the version and format information. After all these operations have been carried out, the decoder tries to detect the color palette. If the color palette is successfully detected, the decoder tries to process a HCC2D code. Otherwise, it tries to decode a QR code. Figure 11 shows the output of an HCC2D code successfully decoded from a HTC Hero mobile phone, running Android 2.1.

We now turn to the experimental results. We executed some performance tests using our prototype. We aimed at measuring the increase in data density and the computational overhead introduced by the use of colors, and thus we measured the time needed to convert all modules in a binary representation according to the color palette considered. Finally we reproduced the Print&Scan process for different print qualities using widely diffused and low-cost print and scan equipment.

The first issue we address is the increase of data density. Like most 2D codes, the QR code data-density depends on several factors. The module size depends heavily on the printing and scanning resolutions. Microsoft offers its HCCB in black and white, with four and with eight different colors. Microsoft laboratory tests have

Fig. 11. A sample screen shot involving a successfully decoded 4-color HCC2D code (version 8). Printout size: 1x1 inches. Device: HTC Hero, running Android 2.1.

yielded data densities of 16,000 bits per square inch with 8 colors in its highest density form using a 600 dpi business card scanner (cfr. [Microsoft Research (2007)]). On the other side, if a QR code symbol is printed with a resolution of 600 dpi, the module size is 0.17mm and will therefore require a scanner resolution of less than 0.17 mm (cfr. [Denso Wave (2011)]). Using Version 19 of the QR code and the M correction level (i.e. about 15%) it is possible to reliably store 5,016 bits per square inch. Introducing a color palette composed of 8 colors, a HCC2D code of Version 19 and with the M correction level, is able to reliably store 15,048 bits per square inch. Those results are summarized in Table 5. Note that the M correction level is an adequate choice for comparing HCCB and HCC2D codes. In fact, HCCB codes can be successfully decoded if the colors of at least 85% of the symbols are correctly identified [Parikh and Jancke (2008)].

Barcode Type	Data Density [bits per square inch]
QR code	5,016
HCC2D	15,048
HCCB	16,000

Table 5. Density of Barcode Symbologies at 600 dpi. The data for HCCB is taken from [Microsoft Research (2007)]

In order to evaluate the usability of HCC2D code in a real setting, we realized a common Print&Scan scenario using a low-quality inkjet multifunction equipment with print and scan capabilities. The equipment is able to print and to scan at different resolutions; in the test scenario we decide to fix the scan resolution (at

600 dpi) while varying the print resolution. We have considered four different print resolutions: Draft Mode (i.e., 180 dpi), Text Mode (i.e., 360 dpi), Text and Photo Mode (i.e., 720 dpi) and Photo (i.e., 1440 dpi). We experimented with small print sizes (fractions of square inches) for different 4-colors HCC2D codes with the M error correction level; the Print&Scan process is repeated an adequate number of times for calculating the success rate. Then, we experimented HCCB codes undergoing the same Print&Scan process. Note that in order to test both HCCB and HCC2D codes under the same conditions, we generated HCCB codes with data densities comparable to those of the HCC2D codes selected. Furthermore, both the correction level and the number of colors were fixed, using 4-colors HCCB and 4-colors HCC2D codes with the same error correction capability. Note that constraints about the HCCB code capacity impact on the experiment design (HCCB is available only in Microsoft Tag format, i.e., a 4-colors HCCB code storing exactly 100 bits). Thus, we had to control the data density parameter varying only the print size and keeping fixed capacities. In our experiments, HCCB and HCC2D codes got similar results. Only codes printed at poor quality levels (i.e., Draft Mode) failed to complete successfully the Print&Scan process. When the data density increases, the success rate depends on the print resolution: at 360 dpi, both HCCB and HCC2D codes were not able to reliably store data at 10,000 $bits/inch^2$.

	180 dpi	360 dpi	720 dpi	1440 dpi
HCCB storing up to 5,000 $bits/inch^2$	None	85%	95%	95%
HCC2D storing up to 5,000 $bits/inch^2$	None	80%	97%	97%
HCCB storing up to 10,000 $bits/inch^2$	None	45%	62%	70%
HCC2D storing up to 10,000 $bits/inch^2$	None	33%	65%	77%

Table 6. Usability of HCCB and HCC2D codes varying the print resolution. The success rate is shown for both HCC2D and HCCB codes, with data densities ranging up to 10,000 $bits/inch^2$.

Now consider that quantitative comparisons between barcodes, based on data density, are not exhaustive. The comparison may be made also in a qualitative manner, thinking about robustness to erasures and damages. Consider the following HCCB codes, in Figure 12. Note that the color palette represents a single point of

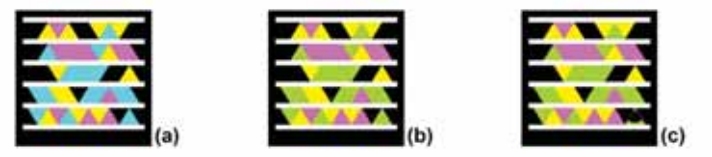


Fig. 12. Hacking HCCB codes: (a) a readable HCCB code, (b) a hacked readable HCCB code (cyan changed to green) and (c) a NOT readable HCCB code (only the last green triangle is damaged). Launch the Microsoft Tag Reader on your phone to test it. (Viewed better in color).

failure. The third code is not readable due to a single damage on the last triangle. HCC2D symbols do not suffer from this concern, replicating the color palette.

We next address the computational overhead required for processing HCC2D codes by comparing the average time taken by the scan process. In Table 7 we report results of an experiment focused on the recognition of more than 100 barcode images. To measure the overhead introduced by colors, times were taken after the alignment and detection phase. The experiment was run on a machine equipped with a Linux Slackware 13.0 operating system running on a 1.73 GHz Intel dual core with 2 GB of RAM. The results report the average time in milliseconds for QR codes, HCC2D with 4 colors and HCC2D with 16 colors, and the overhead (in parenthesis, percentual values) introduced by HCC2D over QR. Our experiments show that, although the overhead introduced by HCC2D over QR tends to increase, as expected, with the number of colors and the code size, it seems to remain always within reasonable values (ranging from a minimum of about 3% for HCC2D version 1H with 4 colors to a maximum of about 38% for HCC2D version 20L with 16 colors). In particular, the average overhead introduced by HCC2D with 4 colors is about 15%, while the average overhead introduced by HCC2D with 16 colors is about 25%.

Version	QR	HCC2D 4 color	HCC2D 16 color
1L	122	132 (7.57%)	135 (9.62%)
1M	123	136 (9.55%)	138 (10.87%)
1Q	129	133 (3.01%)	135 (4.45%)
1H	131	135 (2.97%)	136 (3.68%)
10L	176	209 (15.79%)	246 (28.45%)
10M	188	208 (9.61%)	249 (24.49%)
10Q	189	210 (10.0%)	273 (30.77%)
10H	190	212 (10.37%)	282 (32.62%)
20L	240	349 (31.23%)	387 (37.98%)
20M	253	334 (24.25%)	398 (36.43%)
20Q	250	337 (25.81%)	389 (35.73%)
20H	257	323 (20.43%)	395 (34.93%)
30L	333	447 (25.50%)	474 (29.75%)
30M	350	430 (18.60%)	460 (23.91%)
30Q	347	437 (20.59%)	478 (27.40%)
30H	338	416 (18.75%)	506 (33.20%)
40L	430	483 (10.97%)	550 (21.81%)
40M	415	481 (13.72%)	540 (23.15%)
40Q	420	500 (16.0%)	566 (25.79%)
40H	373	485 (23.09%)	552 (32.42%)

Table 7. Results of the Scan process time in msec for QR and HCC2D

6. Conclusions

In this paper we have proposed HCC2D, High Capacity Colored 2-Dimensional code, a new 2D barcode technology which aims at increasing the data density and at supporting mobile applications. Our results show that the data density of HCC2D is almost close to the one of HCCB, which is considered to be one of the leading barcodes in data density. HCC2D is built on (and is backward compatible with) QR, and thus inherits from QR its strong robustness and error correction properties. In our experiments, HCC2D shows a reasonably small computational overhead with respect to QR and thus seems amenable to practical applications.

There are a number of interesting issues raised by our work, which deserve further investigation. First, it would be nice to carry out a more thorough experimental study of existing 2D and color barcode technologies. This appears to be a difficult task, as some barcode technologies are proprietaries and are not licensed to external parties. Furthermore, we are currently trying to tune HCC2D for other applications, such as the implementation of personal information (e.g., biometric data, sample speech signals, pictures) in cryptographically secure and tamper resistant legal and official documents (e.g., driver licences, boarding documents, passports).

References

Bulan, O., Monga, V. and Sharma, G. High capacity color barcodes using dot orientation and color separability. *Proc. SPIE IS&T Electronic Imaging: Media Forensics and Security*, vol. 7524, 725417-71 (2009).

Denso Wave, Inc. (2011). Quick response code - printer head density and module size. `http://www.denso-wave.com/qrcode/qrgene3-e.html`

Fukuchi, K. (2011). Libqrencode, a C library for encoding data in a QR code symbol. `http://megaui.net/fukuchi/works/qrencode/`. Retrieved January 2011.

Google, Inc. (2011). Zxing, multi-format 1d/2d barcode image processing library. `http://code.google.com/p/zxing/`. Retrieved January 2011.

Grillo, A., Lentini, A., Querini M., Italiano, G.F. (2010) High Capacity Colored Two Dimensional Codes. *Proc. International Multiconference on Computer Science and Information Technology (IMCSIT 2010)*, 2010, pp. 709–716.

Hecht, D.L. (2001a) Printed embedded data graphical user interfaces. *IEEE Computer*, pp. 47–55, 2001.

Hecht, D.L. (2001b) Embedded data glyph technology for hardcopy digital documents. *Proc. SPIE: Color hard copy and graphic arts III,* J. Bares, ed., 2171, pp. 341–352, Mar. 2001.

IATA (2007), `http://www.iata.org/pressroom/pr/Pages/2007-11-10-01.aspx`. Retrieved January 2011.

Jancke, G., Microsoft Research (2011). Personal Communication, January 2011.

Microsoft Research (2007). High Capacity Color Barcodes. `http://research.microsoft.com/projects/hccb/`. Retrieved January 2011.

Parikh, D., Jancke, G. (2008). Localization and Segmentation of a 2D High Capacity Color Barcode. *Proc. 2008 IEEE Workshop on Applications of Computer Vision*, IEEE Computer Society, pp. 1–6.

Siong, K.O., Chai, D., Tan, K.T., (2008). The use of border in colour 2D barcode. *Proc. 2008 International Symposium on Parallel and Distributed Processing with Applications (ISPA 2008)*, pp. 999–1005.