QR CODE

An industrial application of Code Theory

Davide Boscaini, Simone Parisotto

Università di Verona
Dipartimento di Scienze Matematiche Fisiche e Naturali
-
Laurea Magistrale in Matematica
Corso di Algebra Computazionale

18 Ottobre 2012

History

- QR-Code is a two dimensional barcode (datamatrix);
- The acronym QR is derived from the term *Quick Response*;
- Created to store more data and characters than classical barcodes;
- Invented in Toyota subsidiary *Denso Wave* in 1994 to track vehicles during the manufacturing process;
- Japanese standard for QR-Codes, devised by Denso Wave, is JIS X 0510 (January 1999). ISO International Standard (ISO/IEC 18004), approved in June of 2000, updated back in 2006 (ISO/IEC 18004:2006);
- Today applications: Magazines, Papers, Business Cards, Buses, Signs, T-shirts...


Developers still waiting for tenders to redevelop RRI land in Sungai Buloh

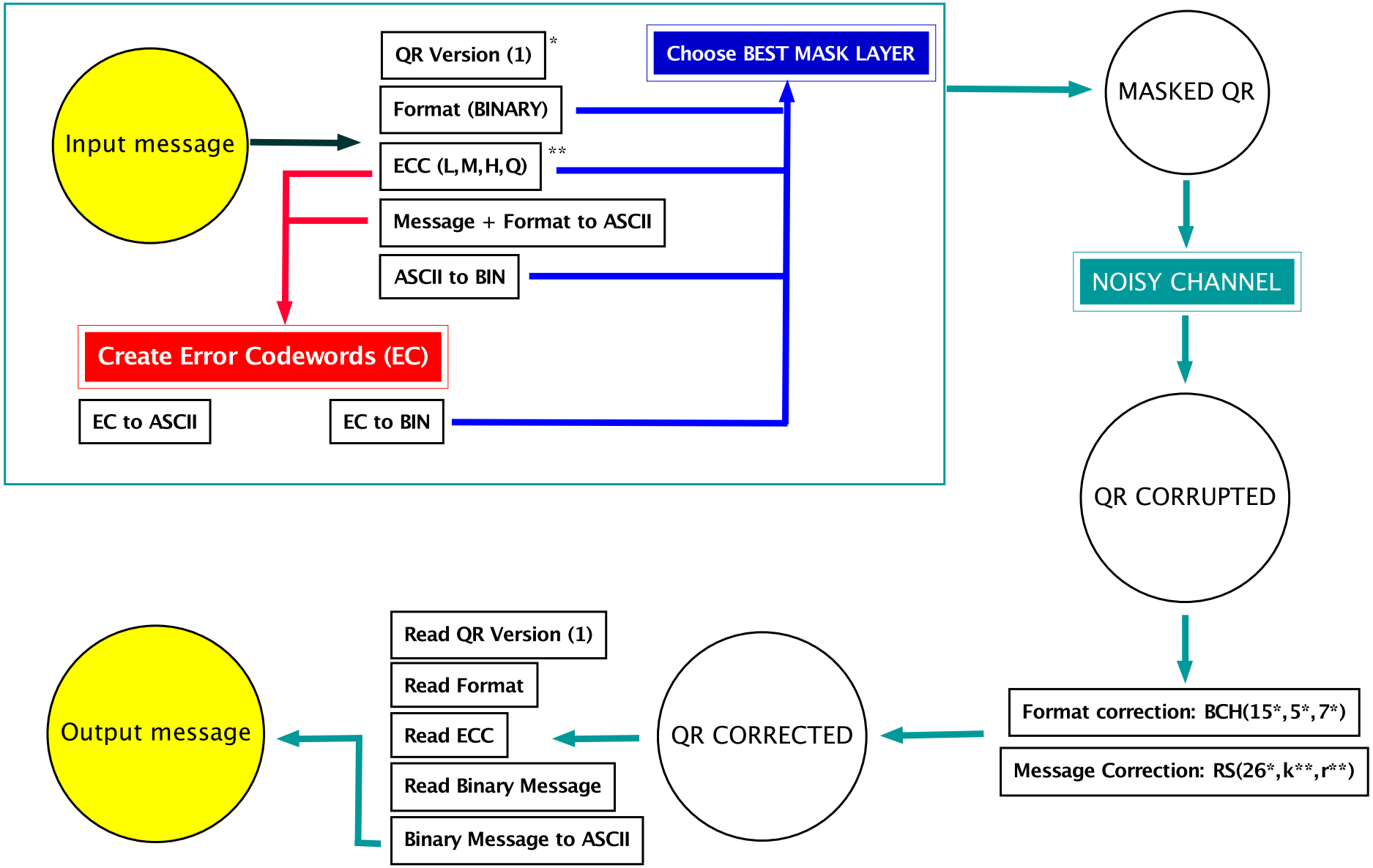
SHARE THIS STORY WITH QPAPER

yet to hear of more details or any formal time line being set.

Said a developer: "We have not been

> TURN TO PAGE 2

Workflow


Input message
QR Version (1) *
Format (BINARY)
ECC (L,M,H,Q) **
Message + Format to ASCII
ASCII to BIN
Create Error Codewords (EC)
EC to ASCII
EC to BIN
Choose BEST MASK LAYER
MASKED QR
NOISY CHANNEL
QR CORRUPTED
Format correction: BCH(15*,5*,7*)
Message Correction: RS(26*,k**,r**)
QR CORRECTED
Read QR Version (1)
Read Format
Read ECC
Read Binary Message
Binary Message to ASCII
Output message

Case study: *'Twas brillig* (Lewis Carroll, 1871) - QR 1

Input Type			ECC			Blocks Number	
Bin	ID	Bit Length	Bin	ID	%	Message	EC
[0001]	N	10	[01]	L	7%	19	7
[0010]	A	9	[00]	M	15%	16	10
[0100]	B	8	[11]	Q	25%	13	13
[1000]	K	8	[10]	H	30%	9	17

	Type	Length	Message	EOF	Extra (236,17)
	Byte	13	'Twas brillig	0000	$16 - 2 - 13 = 1$
ASCII			39 84 119 97 115 32 98 114 105 108 108 105 103		236
BIN	0100	00001101	00100111 01010100 01110111 01100001 01110011 00100000 01100010 01110010 01101001 01101100 01101100 01101001 01100111	0000	11101100
BIN	01000000 00100111	11010010 00100110	01110101 01000111 01110110 00010111 00110010 00000110 10010110 11000110 11000110 10010110 01110000 11101100		
ASCII			64 210 117 71 118 23 50 6 39 38 150 198 198 150 112 236		
POLY			$m(x) = 64x^{15} + 210x^{14} + 117x^{13} + 71x^{12} + 118x^{11} + 23x^{10} + 50x^{9} + 6x^{8} +$ $+39x^{7} + 38x^{6} + 150x^{5} + 198x^{4} + 198x^{3} + 150x^{2} + 112x + 236$		

Creating Error Codewords

- The number of erasures and errors correctable is given by the following formula:
  $e + 2t \leq d - p$, where
  - e = number of erasures (erroneous codewords at known locations): unscanned or undecodable symbol character;
  - t = number of errors (erroneous codewords at unknown locations): misdecoded symbol character;
  - d = number of error correction codewords;
  - p = number of misdecode protection codewords;

ECC	d	p	t			RS(c,k,r)**	Recovery Capacity
			$e = 0$	$e = 1$	$e = 2$		
L	7	3*	2	0	0	(26,19,2)	$100 * r/26 \approx 07\%$
M	10	2*	4	3	2	(26,16,4)	$100 * r/26 \approx 15\%$
Q	13	1*	6	5	4	(26,13,6)	$100 * r/26 \approx 25\%$
H	17	1*	8	7	6	(26,9,8)	$100 * r/26 \approx 30\%$

(*) for QR Version 1

(**) c = total number of codewords, k = number of data codewords, r = error correction capacity

Generator polynomials & Encoding Error Codewords

Number of error correction codewords	Generator polynomials, $C = (g(x))$ $g(x) = (x - \alpha^{0})(x - \alpha^{1}) \dots (x - \alpha^{(n-k-1)})$
L: $n - k = 07$	$x^{7} + \alpha^{87}x^{6} + \alpha^{229}x^{5} + \alpha^{146}x^{4} + \alpha^{149}x^{3} + \alpha^{238}x^{2} + \alpha^{102}x + \alpha^{21}$;
M: $n - k = 10$	$x^{10} + \alpha^{251}x^{9} + \alpha^{67}x^{8} + \alpha^{46}x^{7} + \alpha^{61}x^{6} +$ $+\alpha^{118}x^{5} + \alpha^{70}x^{4} + \alpha^{64}x^{3} + \alpha^{94}x^{2} + \alpha^{32}x + \alpha^{45}$;
Q: $n - k = 13$	$x^{13} + \alpha^{74}x^{12} + \alpha^{152}x^{1}1 + \alpha^{176}x^{10} + \alpha^{100}x^{9} + \alpha^{86}x^{8} + \alpha^{100}x^{7} +$ $+\alpha^{106}x^{6} + \alpha^{104}x^{5} + \alpha^{130}x^{4} + \alpha^{218}x^{3} + \alpha^{206}x^{2} + \alpha^{140}x + \alpha^{78}$;
H: $n - k = 17$	$x^{17} + \alpha^{43}x^{16} + \alpha^{139}x^{15} + \alpha^{206}x^{14} + \alpha^{78}x^{13} + \alpha^{43}x^{12} +$ $+\alpha^{239}x^{11} + \alpha^{123}x^{10} + \alpha^{206}x^{9} + \alpha^{214}x^{8} + \alpha^{147}x^{7} + \alpha^{24}x^{6} +$ $+\alpha^{99}x^{5} + \alpha^{150}x^{4} + \alpha^{39}x^{3} + \alpha^{243}x^{2} + \alpha^{163}x + \alpha^{136}$;
α is the primitive element 2 under GF(2^{8});	

We chose M as ECC ID for *'Twas brillig* so we expect deg $r(x) = 9$ in $x^{n-k}m(x) = a(x)g(x) + r(x)$, where

- $a(x) = 64x^{15} + 214x^{14} + 88x^{13} + 145x^{12} + 17x^{11} + 169x^{10} + 127x^{9} + 62x^{8} + 105x^{7} + 248x^{6} + 96x^{5} + 35x^{4} + 97x^{3} + 244x^{2} + 151x + 18$;
- $r(x) = 188x^{9} + 42x^{8} + 144x^{7} + 19x^{6} + 107x^{5} + 175x^{4} + 239x^{3} + 253x^{2} + 75x + 224$.

Coefficients of $r(x)$ are our error correction codewords.
So QR Code must include (in binary) message codewords with error correction codewords:
[64, 210, 117, 71, 118, 23, 50, 6, 39, 38, 150, 198, 198, 150, 112, 236, 188, 42, 144, 19, 107, 175, 239, 253, 75, 224].

Finite field arithmetic

- GF(p^n) is the finite field with p^n element (p prime): the ring of integers modulo p;
- elements of GF(p^n) are represented as polynomials (degree $< n$) over GF(p);
- operations are performed modulo R, an irreducible polynomial of degree n over GF(p);
- if $p = 2$, the elements of GF(p^n) are expressed as binary numbers.

Addition and Subtraction

Performed by adding or subtracting two of these polynomials together, and reducing the result modulo the characteristic. Example in GF(2): $(x^3 + x + 1) + (x^3 + x^2) = x^2 + x + 1$, $(\neq 2x^3 + x^2 + x + 1)$.

Multiplication

Is multiplication modulo an irreducible reducing polynomial used to define the finite field. Example with irreducible reducing polynomial $R(x) = x^8 + x^4 + x^3 + x + 1$:
$(x^6 + x^4 + x + 1)(x^7 + x^6 + x^3 + x) = (x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x) \mod R(x) =$
$= (11111101111110 \mod 100011011) = 1$ (demostrated with long division with XOR).

Division (aka Multiplicative Inverse)

Is performed making a logarithm table of the finite field, and performing subtraction in the table. Subtraction of logarithms is the same as division. Example: $8 : 4 = 2$ is equal to $\log(\alpha^3) - \log(\alpha^2) = 1$ and $\alpha^1 = 2$ with 2 primitive element of GF(2^8).

Why Reed-Solomon?

Reed-Solomon are a good choice because:

- are useful in correcting *burst* error (used in concatenated form): CD, DVD ...;
- they are optimal $k = n - d + 1$ (Maximum Separable Distance);

but they require a large alphabet size (Singleton Bound).

If we operate on bits, how to convert the codewords over the large field in the binary alphabet?

Example: write every element of a code defined over $\mathbb{F}_{256}$ as an 8-bit vector.

Theory of *Concatenated* and *Shortened* Reed-Solomon Codes is applied in QR Code. But the resulting code isn't *optimal*: BCH is better!

Answer to the question: Why is still used Reed-Solomon instead of BCH?

The main reason that Reed-Solomon are still frequently used is that in many applications – and in particular in storage device applications – errors often occur in bursts. Reed-Solomon codes have the nice property that bursts of consecutive errors affect bits that correspond to a much smaller number of elements in the field on which the Reed-Solomon code is defined.

Example: if a binary code constructed from the RS(256,230) code is hit with 30 consecutive errors, these errors affect at most 5 elements in the field $\mathbb{F}_{256}$ and this error is easily corrected.

Concatenated and Shortened Reed-Solomon codes

Concatenated RS-codes (Forney, 1966)

Let $\mathcal{A}(n,k,d)$ the inner-code over $\mathbb{F}_q$. Let $Q = q^k$ and define $\psi : \mathbb{F}_Q \to \mathcal{A}$ a one-to-one $\mathbb{F}_q$-linear map. $\mathbb{F}_Q$ is an extension field of $\mathbb{F}_q$. Let $\mathcal{B}$ an (N,K,D) outer-code over $\mathbb{F}_Q$. The *concatenation* of $\mathcal{A}$ and $\mathcal{B}$ is the code $\mathcal{C} = \{\psi(b_1, b_2, \ldots, b_N) | (b_1, b_2, \ldots, b_N) \in \mathcal{B}\}$ where $\psi(b_1, b_2, \ldots, b_N) = (\psi(b_1), \psi(b_2), \ldots, \psi(b_N))$.

Theorem

Let $\mathcal{A}$ and $\mathcal{B}$ as above. Then $\mathcal{C}$ is a linear (nN, kK) code over $\mathbb{F}_q$, (minimum distance $\geq d \cdot D$).

Example: In our case study we have $\mathcal{A}(8,8,1)$ and $\mathcal{B}(26,16,9*)$ (*with misdecode protection codewords). $\mathcal{C}$ is a linear $(8 \cdot 26, 8 \cdot 16) = (208, 128)$ code over $\mathbb{F}_2$ (min. distance $\geq 1 \cdot 9 = 9$).

Shortened RS-codes

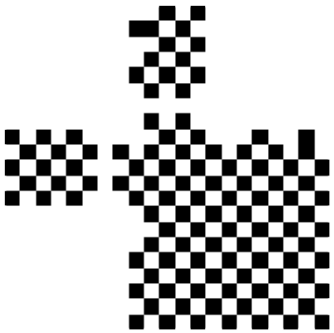
Reed-Solomon codes may be shortened by (conceptually) making a number of data symbols zero at the encoder, not transmitting them, and then re-inserting them at the decoder.

Example: A (256,175) code can be shortened to (208,128). The encoder takes a block of 128 data bits, (conceptually) adds 48 zero bits, creates a (256,175) codeword and transmits only the 128 data bits and 80 parity bits.

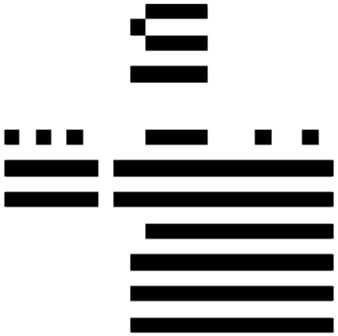
NB: Each generator $g(x)$ provided from the QR ISO standard divide $x^{256} - 1$.

Choosing the Best Mask Layer

- i,j start from 0;
- % is the modulo operation, $\div$ is the integer division;

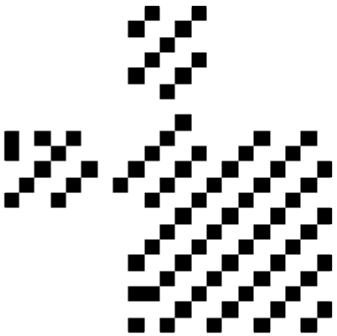


000
$(i+j)\%2=0$

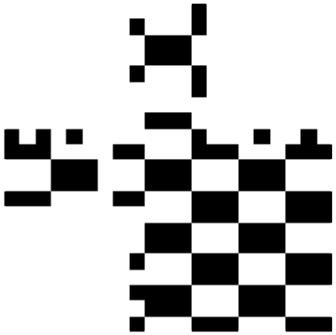


001
$i\%2=0$

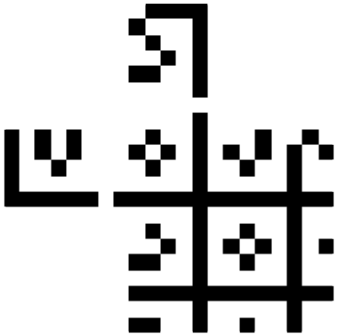
010
$j\%2=0$



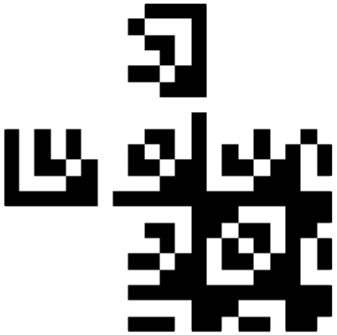
011
$(i+j)\%3=0$



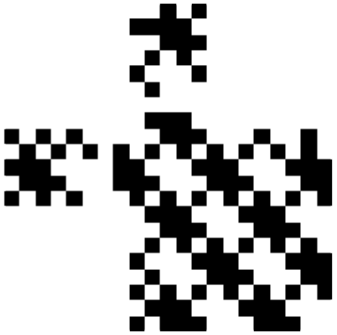
100
$((i \div 2)+(j \div 3))\%2=0$



101
$(ij)\%2+(ij)\%3=0$



110
$((ij)\%2+(ij)\%3)\%2=0$



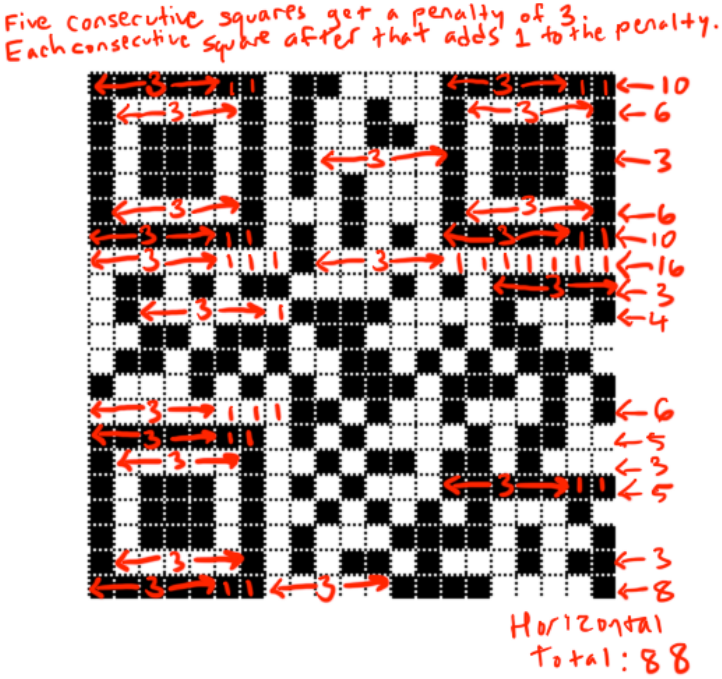
111
$((i+j)\%2+(ij\%3))\%2$

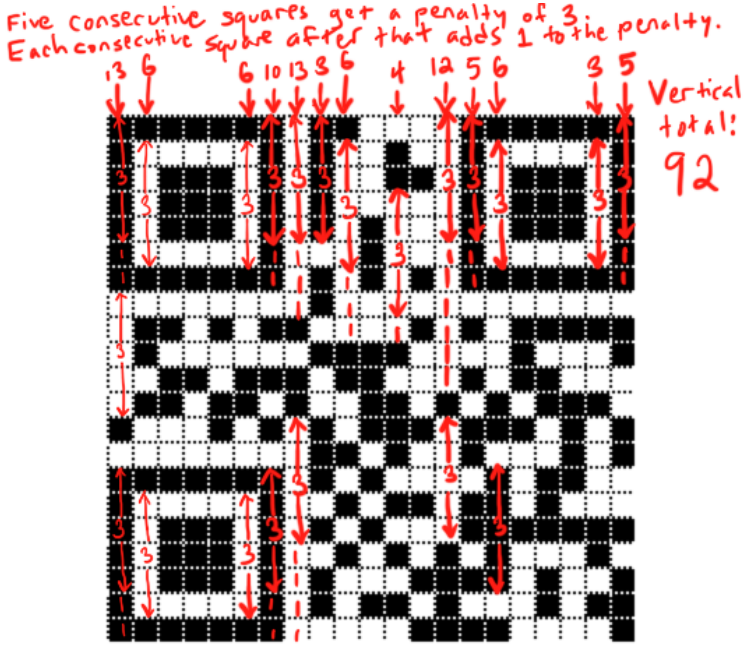
Penalty Method

Each unmasked qrcode must be masked for a better recognition in the device scanner. Based on a penalty method, we evaluate each masked qrcode we choose the mask layer with the lowest penalty. An example to follow.

Penalty rule #1

If five or more of the same colored pixels are next to each other in a row or column. For the first five consecutive pixels, the penalty score is increased by 3. Each consecutive pixel after that adds 1 to the penalty.





Penalty rule #2

Each 2x2 block of the same color adds a penalty of 3 to the amount.

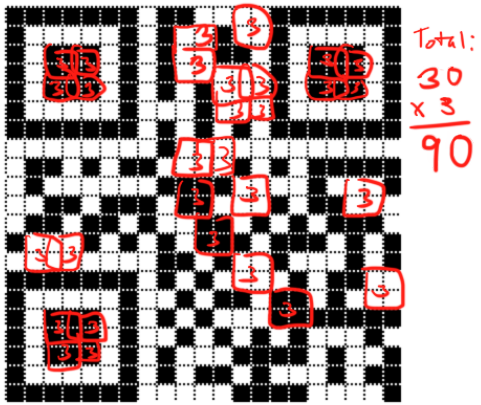
Penalty rule #3

Each pattern (in row or column) [1 0 1 1 1 0 1] with 4 white pixels on either or both sides adds a penalty of 40 to the amount.

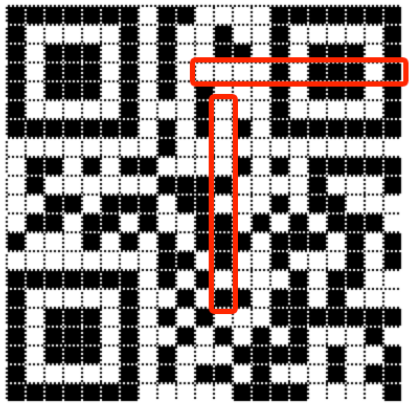
Penalty rule #4

This rule is based on the ratio of dark to light pixels: the closer the ratio is to 50% dark and 50% light, the better the penalty score will be.
Formula: $10 * \text{abs}(\text{fix}(100 * (\#\text{black pixels}/\#\text{total pixels}) - 50))/5$

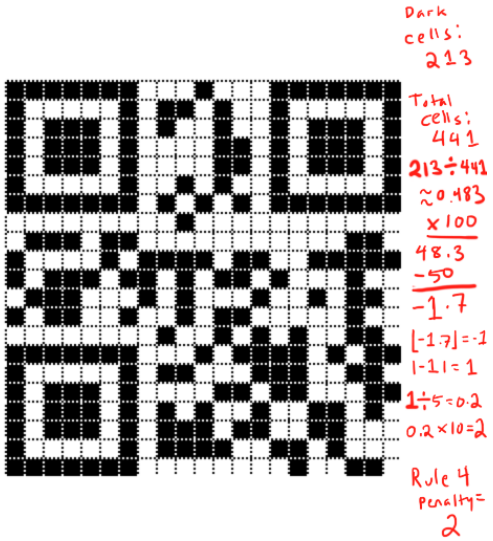
Penalty rule #2



Penalty rule #3

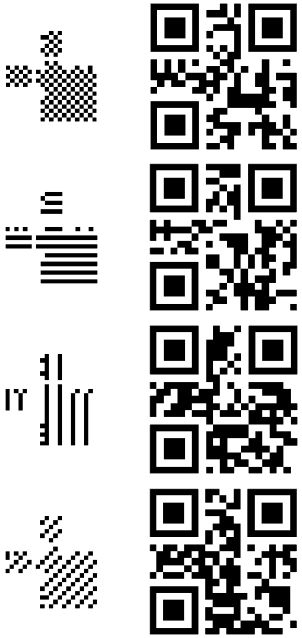


Penalty rule #4



Penalty results for our case study

Here we present the penalty method applied to our unmasked QR image:

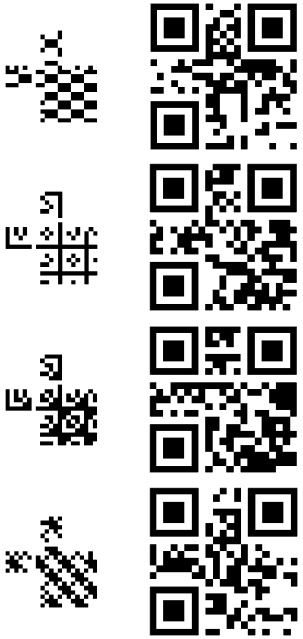


mask 0: p1+p2+p3+p4 = 452

mask 1: p1+p2+p3+p4 = 445

mask 2: p1+p2+p3+p4 = 337

mask 3: p1+p2+p3+p4 = 367



mask 4: p1+p2+p3+p4 = 436

mask 5: p1+p2+p3+p4 = 511

mask 6: p1+p2+p3+p4 = 408

mask 7: p1+p2+p3+p4 = 549

So we choose mask layer ID 2 which pattern is 0 1 0.

Encode Format Row and write the image

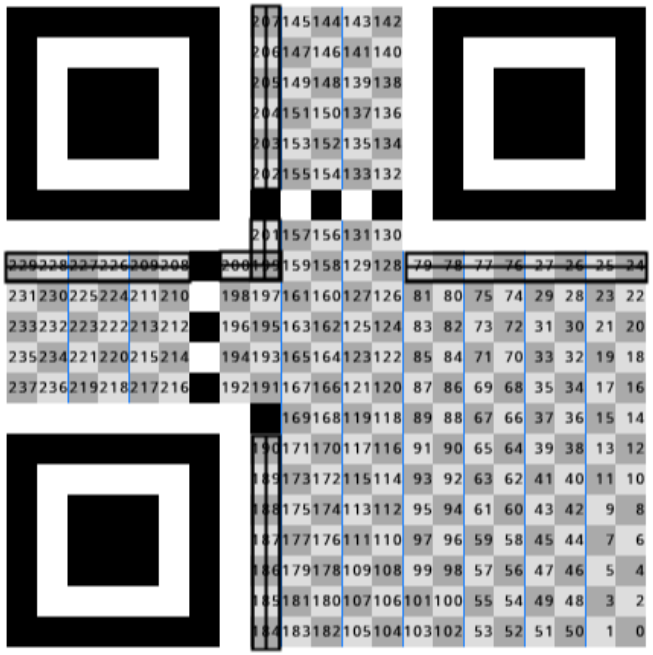
The penalty method is applied to every mask choosen in order to apply he best mask layer. What about encoding information row? Choosing

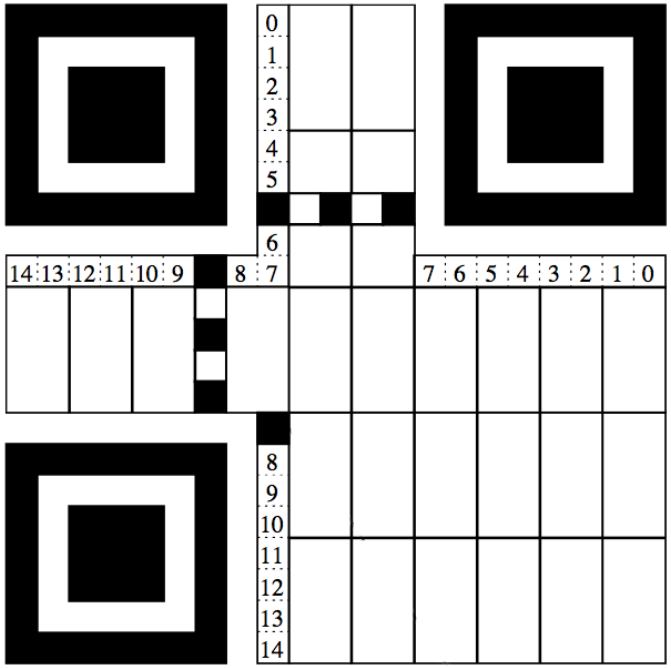
$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$$

as generator polynomial for BCH code (15,5), we have

$$ax^{14} + bx^{13} + cx^{12} + dx^{11} + ex^{10} | g(x),$$

with a, b coefficients of the ECC ID choosen and c, d, e the mask layer ID choosen. The remainder of this operation is what to include in the last 10 empty bits. So we can write the encoded message in the image:



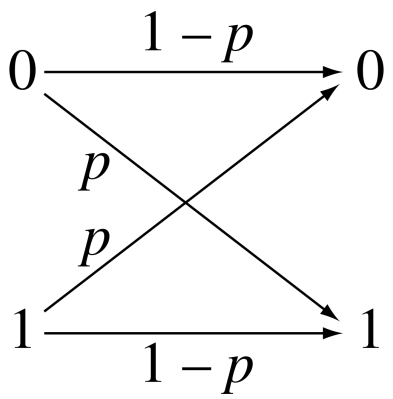


The Noisy Channel

Now we have to create a mathematical model of a noisy transmission channel. A possible mathematical model is **Binary Symmetric Channel** (BSC): if 0 or 1 is sent, the probability that it is received without error is $1-p$; if a 0 (respectively 1) is sent, the probability that a 1 (respectively 0) is received is p. So the probability that one bit is received without error is $1-p$, and then the probability that is received the wrong bit is p. In most practical situations p is very small. A BSC has capacity

$$C(p) = 1 + p\log_2 p + (1-p)\log_2(1-p).$$

The following illustration describes quite well the precedent model



In our case study we choose a probability $p = 0.15$ and then $C(p) \approxeq 0.4$.

Shannon Theorem

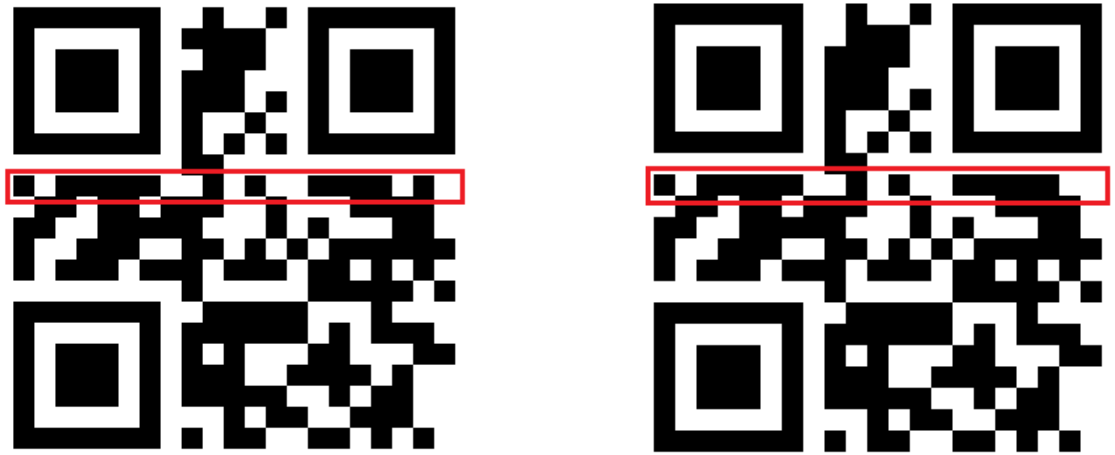
Given $\delta > 0$ and $R < C(p)$ exist a linear binary code C(n,k) with $k/n \geq R$ and $\mathrm{P_{err}} = 1 - \sum_{i=0}^{n} \alpha_i p^i (1-p)^{n-i} < \delta$ (with α_i number of cosets of weigth i).

For implement this mathematical model we use the MATLAB function `bsc`,

```
NDATA = bsc(DATA,P) passes the binary input signal DATA through a binary
symmetric channel with error probability P. If the input DATA is a Galois field
over GF(2), the Galois field data is passed through the binary symmetric
channel.
NDATA = bsc(DATA,P,S) causes RAND to use the random stream S. S is any valid
random stream.
```

where we have fixed the seem of the random number generator to compare repeated experiments.

We choose to pass trough the BSC only the format pattern. The result is the following



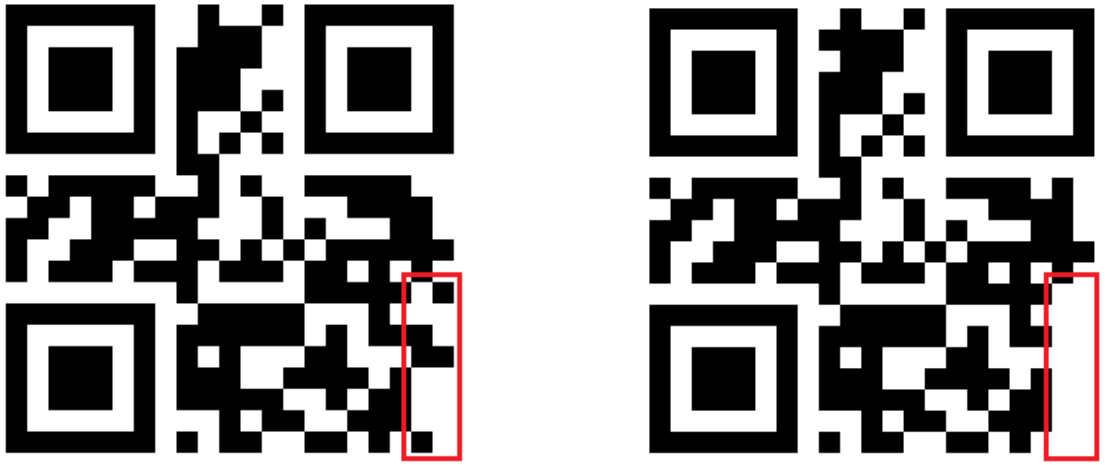
In the left figure is highlighted the format pattern of the original QR code, in the right one is highlighted the format pattern of the received QR code.

For the rest of the code, i.e. the message codewords and the error correcting codeword, the effect of noise in the transmission channel is manifested by the occurrence of errors in codewords. In our test study we suppose that the scanner of the code can't read in the correct way the first two codewords. The effect of the wrong lecture is that all the bits of the first two codewords are zeros.

Observation

We consider this an error and not an erasure.

The result is the following



In the left figure there is the original QR code, in the right one the received QR code.

BCH(15,5,7) for correcting information pattern

The information pattern is a BCH(15,5,7) code. BCH code were discovered around 1960 by Hocqueghem and independently by Bose and Ray-Chadhuri. For the description of this algorithm we mainly refer to the section 5.1 of the book *Fundamentals of Error Correcting Codes* wrote by C. Huffman and V. Pless.

BCH codes are cyclic codes designed to take advantage of the BCH bound, i.e.

Theorem (BCH Bound)

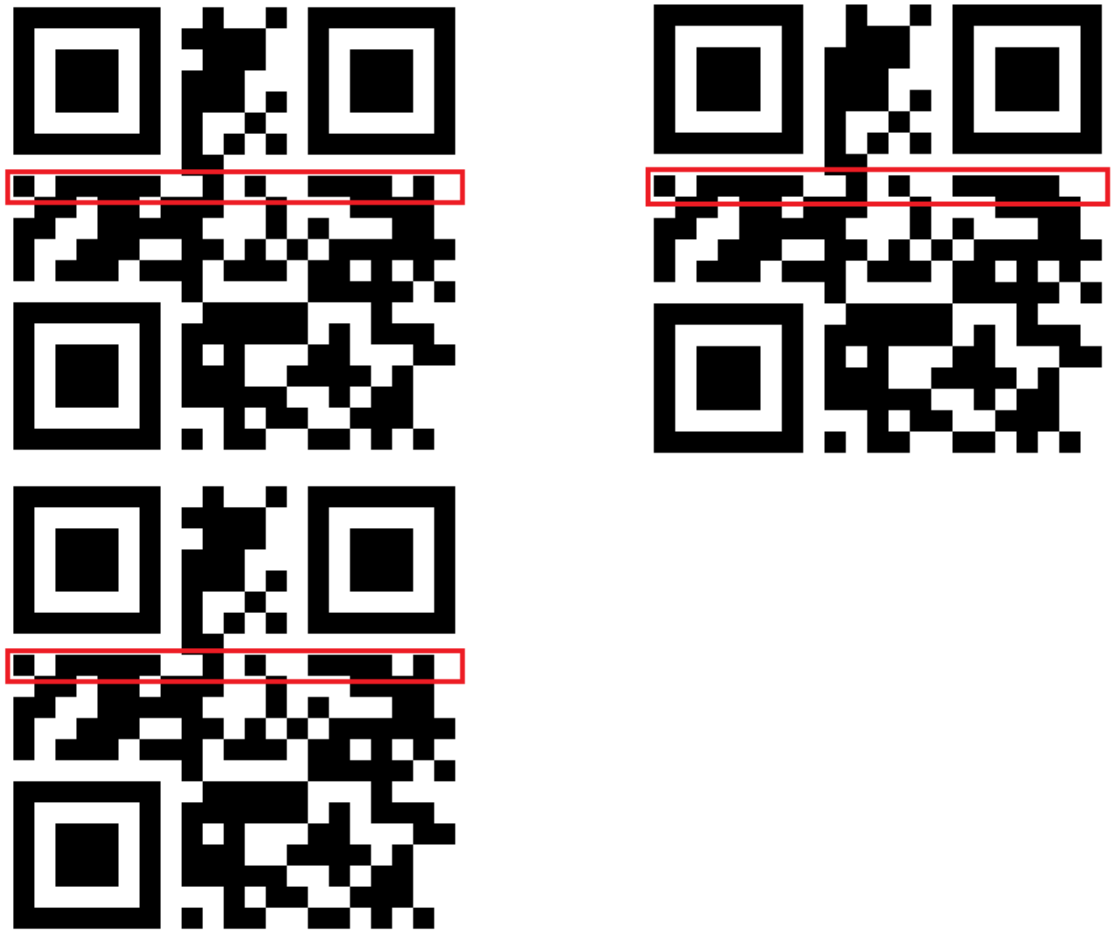
Let $\mathcal{C}$ be a cyclic code of length n over $\mathbb{F}_q$ with defining set T. Suppose $\mathcal{C}$ has minimum weight d. Assume T contains $\delta - 1$ consecutive elements for some integer δ. Then $d \geq \delta$.

For decoding this code we use the MATLAB function `bchdec`,

```
DECODED = bchdec(CODE,N,K) attempts to decode the received signal in CODE using
an (N,K) BCH decoder with the narrow-sense generator polynomial.
CODE is a Galois array of symbols over GF(2).
Each N-element row of CODE represents a corrupted systematic codeword, where the
parity symbols are at the end and the leftmost symbol is the most significant
symbol.  bchdec uses the Berlekamp-Massey decoding algorithm.
```

In our case study we have $\mathtt{N} = 15$ and $\mathtt{K} = 5$.

The result is the following



where the top left figure represent the information pattern of original QR code, the top right the information pattern of the QR code received and the bottom left the restored information pattern.

The PGZ Algorithm for correcting QR Symbol

As we have seen in the previous slides, Reed-Solomon codes are a particular subfamily of BCH codes. We choose to decoding them with the **Peterson-Gorenstein-Zierler Algorithm**. This method was originally developed for binary codes by Peterson in 1960 and generalized shortly thereafter by Gorenstein and Zierler to nonbinary BCH codes (our case study). For the description of this algorithm we mainly refer to the section 5.4 of the book *Fundamentals of Error Correcting Codes* wrote by C. Huffman and V. Pless.

Let $\mathcal{C}$ be a BCH code over $\mathbb{F}_q$ of length n and designed distance δ. As the minimum distance of $\mathcal{C}$ is at least δ, $\mathcal{C}$ can correct at least $t = \lfloor(\delta - 1)/2\rfloor$ errors. The PGZ Decoding Algorithm will correct up to t errors. Therefore the defining set T of $\mathcal{C}$ will be assumed to contain $\{1, 2, \ldots, \delta - 1\}$, with α the primitive nth root of unity in the extension field $\mathbb{F}_{q^m}$ of $\mathbb{F}_q$, where $m = \mathrm{ord}_n(q)$.

Suppose that $y(x)$ is received and that it differs from a codeword $c(x)$ in at most t coordinates. Therefore $y(x) = c(x) + e(x)$ where $c(x) \in \mathcal{C}$ and $e(x)$ is the *error vector* witch has weight $\nu \leq t$. Suppose that the errors occur in the unknown coordinates $k_1, k_2, \ldots, k_\nu$. Therefore

$$e(x) = e_{k_1} x^{k_1} + e_{k_2} x^{k_2} + \cdots + e_{k_\nu} x^{k_\nu}. \tag{1}$$

Once we determine $e(x)$, which amounts to finding the error locations k_j and the error magnitudes e_{k_j}, we can decode the received vector as $c(x) = y(x) - e(x)$.

Recall that $c(x) \in \mathcal{C}$ if and only if $c(\alpha^i) = 0$ for all $i \in T$. In particular

$$y(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i) \text{ for all } 1 \leq i \leq 2t,$$

since $2t \leq \delta - 1$.

The PGZ decoding algorithm requires four steps.

First step

Compute the syndromes $S_i = y(\alpha^i)$ for $1 \leq i \leq 2t$ from the received vector (we are working with the arithmetic of the finite field $\mathbb{F}_{q^m}$).

For this step is quite useful the following

Theorem

$S_{iq} = S_i^q$ for all $i \geq 1$.

because it allows us to avoid a lot of evaluations of the received polynomial and then could help us to reduce the computation costs of the algorithm.

Notice that from the equation (1) the syndromes satisfy

$$S_i = y(\alpha^i) = \sum_{j=1}^{\nu} e_{k_j}(\alpha^i)^{k_j} = \sum_{j=1}^{\nu} e_{k_j}(\alpha^{k_j})^i$$

for $1 \leq i \leq 2t$. To simplify the notation, for $1 \leq j \leq \nu$, let $E_j = e_{k_j}$ denote the *error magnitude at coordinate* k_j and $X_j = \alpha^{k_j}$ denote the *error location number corresponding to the error location* k_j. With this notation become

$$S_i = \sum_{j=1}^{\nu} E_j X_j^i, \text{ for } 1 \leq i \leq 2t, \tag{2}$$

which leads to the system of equations

$$\begin{aligned} S_1 &= E_1 X_1 + E_2 X_2 + \cdots + E_\nu X_\nu \\ S_2 &= E_1 X_1^2 + E_2 X_2^2 + \cdots + E_\nu X_\nu^2 \\ &\vdots \\ S_{2t} &= E_1 X_1^{2t} + E_2 X_2^{2t} + \cdots + E_\nu X_\nu^{2t}. \end{aligned} \tag{3}$$

This system is obviously nonlinear in the X_js with unknown coefficients E_j.

The strategy is to transform the precedent into a linear system involving new variables $\sigma_1, \sigma_2, \ldots, \sigma_\nu$, that will lead directly to the error location numbers. Once these are known, we return to the system (3), witch is then a linear system in the E_js and solve for the error magnitudes.
To this end, define the *error locator polynomial* to be

$$\sigma(x) = (1 - xX_1)(1 - xX_2)\cdots(1 - xX_\nu) = 1 + \sum_{i=1}^{\nu} \sigma_i x^i.$$

The roots of $\sigma(x)$ are the inverses of the error location numbers and thus

$$\sigma(X_j^{-1}) = 1 + \sigma_1 X_j^{-1} + \sigma_2 X_j^{-2} + \cdots + \sigma_\nu X_j^{-\nu} = 0 \text{ for } 1 \leq j \leq \nu. \tag{4}$$

Multiplying (4) by $E_j X_j^{i+\nu}$ produces

$$E_j X_j^{i+\nu} + \sigma_1 E_j X_j^{i+\nu-1} + \cdots + \sigma_\nu E_j X_j^{i} = 0 \text{ for any } i.$$

Summing the result obtained over j for $1 \leq j \leq \nu$ yields

$$\sum_{j=1}^{\nu} E_j X_j^{i+\nu} + \sigma_1 \sum_{j=1}^{\nu} E_j X_j^{i+\nu-1} + \cdots + \sigma_\nu \sum_{j=1}^{\nu} E_j X_j^{i} = 0$$

As long as $1 \leq i$ and $i + \nu \leq 2t$, these summations are the syndromes obtained in (2). Because $\nu \leq t$, the precedent equation becomes

$$\sigma_1 S_{i+\nu-1} + \sigma_2 S_{i+\nu-2} + \cdots + \sigma_\nu S_i = -S_{i+\nu} \text{ for } 1 \leq i \leq \nu.$$

Thus we can find the σ_ks if we solve the matrix equation

$$\begin{bmatrix} S_1 & S_2 & \dots & S_{\nu-1} & S_\nu \\ S_2 & S_3 & \dots & S_\nu & S_{\nu+1} \\ \vdots & \vdots & & \vdots & \vdots \\ S_\nu & S_{\nu+1} & \dots & S_{2\nu-2} & S_{2\nu-1} \end{bmatrix} \begin{bmatrix} \sigma_\nu \\ \sigma_{\nu-1} \\ \vdots \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_{\nu+1} \\ -S_{\nu+2} \\ \vdots \\ -S_{2\nu}. \end{bmatrix} \tag{5}$$

Lemma

Let $\mu \leq t$ and let

$$M_\mu = \begin{bmatrix} S_1 & S_2 & \dots & S_\mu \\ S_2 & S_3 & \dots & S_{\mu+1} \\ \vdots & \vdots & & \vdots \\ S_\mu & S_{\mu+1} & \dots & S_{2\mu-1} \end{bmatrix}.$$

Then M_μ is nonsingular if $\mu = \nu$ and singular if $\mu > \nu$, where ν is the number of errors that have occurred.

To execute the second step of our algorithm, we attempt to guess the number ν of errors. Call our guess μ and starts with $\mu = t$, witch is the largest that ν could be. The coefficients matrix of the linear system (5) is $M_\mu = M_t$.

Second step

In the order $\mu = t$, $\mu = t - 1$, ... decide if M_μ is singular, stopping at the first value of μ where M_μ is nonsingular. Set $\nu = \mu$ and solve (5) to determine $\sigma(x)$.

Third step

Find the roots of $\sigma(x)$ by computing $\sigma(\alpha^i)$ for $0 \leq i < n$. Invert the roots to get the error location number X_j.

Fourth step

Solve the first ν equations of (3) to obtain the error magnitudes E_j.

In fact we need to consider only the first ν equations in (3) because the coefficient matrix of the first ν equations has determinant

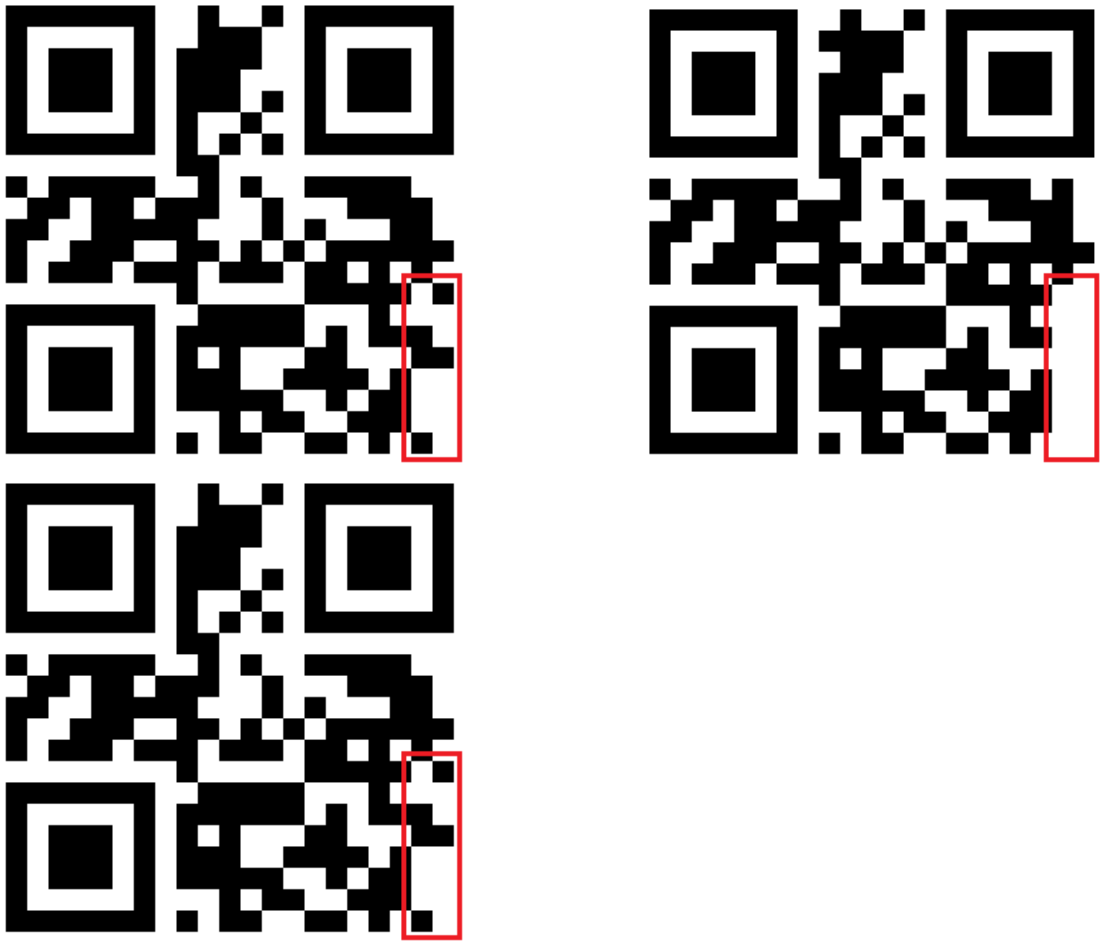
$$\det \begin{bmatrix} X_1 & X_2 & \dots & X_\nu \\ X_1^2 & X_2^2 & \dots & X_\nu^2 \\ \vdots & \vdots & & \vdots \\ X_1^\nu & X_2^\nu & \dots & X_\nu^\nu \end{bmatrix} = X_1 X_2 \cdots X_\nu \det \begin{bmatrix} 1 & 1 & \dots & 1 \\ X_1 & X_2 & \dots & X_\nu \\ \vdots & \vdots & & \vdots \\ X_1^{\nu-1} & X_2^{\nu-1} & \dots & X_\nu^{\nu-1} \end{bmatrix}$$

The latter is the transpose of a Vandermonde matrix and is well known that its determinant is nonzero as the X_js are distinct.

Observation

If the BCH code is binary, all error magnitudes must be 1. Hence step four can be skipped.

The result is the following



where the top left figure represent the original QR code, the top right the received QR code and the bottom left the restored QR code.

How can PGZ algorithm be improved?

The title of this slide report a question that couldn't be skipped.

The second step of the PGZ Algorithm is the most complicated and time consuming. In this step in fact we have to solve the linear system (5), a problem that corresponds to the inversion of the matrix M_μ.

This isn't a problem when the error capability of the code is rather small, because, in these cases, the matrix M_μ has small dimensions and the PGZ Algorithm is quite efficient. But when the error capability of the code is very large and then the size of the matrices M_μ becomes very large, the inversion of the matrix M_μ become an hard problem and step two becomes very time consuming.

We can prevent this problem choosing one of the following algorithms

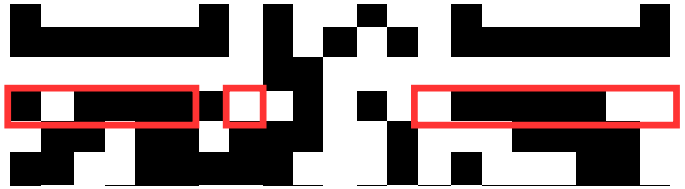
- The **Berlekamp-Massey Algorithm** uses an iterative approach to compute the error locator polynomial in a more efficient manner when t is large.
- The **Sugiyama Algorithm** is another method that uses the Euclidean Algorithm to find the error locator polynomial. This algorithm is quite comparable in efficiency with the Berlekamp-Massey Algorithm.

Finally, also step three can be quite time consuming if the code is long, however little seems to have been done to improve this step.

Recovery hidden output message

Once we have corrected the corrupted image, we are ready to recovery the hidden message.

- Select the format information row: [1 0 1 1 1 1 0 0 1 1 1 1 1 0 0];



- Unmask the first 5 bits [1 0 1 1 1] with the standard rule: [1 0 1 1 1] − [1 0 1 0 1] = [0 0 0 1 0];
- [0 0] is the ECC format recognized: M is the ECC;
- [0 1 0] is the mask layer's format recognized: 2 is its mask layer ID.

Then we can discover the hidden image:

Masked Image

−

Mask Layer

=

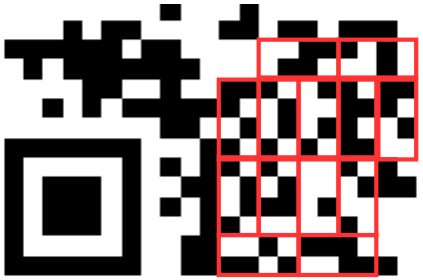
Unmasked Image

Once obtained the unmasked image is very simple to read the hidden message.

- Check the 2×2 block in the right-bottom corner of the image:

- Unroll it: [0 1 0 0]. It tells us the message is in Binary format;
- Check the 4×2 block on top of previous block:

- Unroll it: [0 0 0 0 1 1 0 1]. It tell us there are 13 codewords to read;
- Remembering that 1 Byte is 8 Bit for M, read the next 13*8 bits:



  or, in decimal: [39 84 119 97 115 32 98 114 105 108 108 105 103];
- The previous sequence, converted in unicode, returns *'Twas brillig*.

Our decoding process ends successfully.

Extra Tests

Some QR images generated from our MATLAB code:

www.univr.it

Hello World

id000000@univr.it

Bibliography

- *Fundamentals of Error Correcting Codes*, C. Huffman, V. Pless, Cambridge U. Press;
- Information technology, *Automatic identification and data capture techniques, QR Code 2005 bar code symbology specification*, INTERNATIONAL ISO/IEC STANDARD 18004
- http://www.pclviewer.com/rs2/calculator.html
- http://www.thonky.com/qr-code-tutorial/

Please take care to use error correction codewords from the last two website: the polynomial division algorithm fails in some cases!

Thank you for your attention.