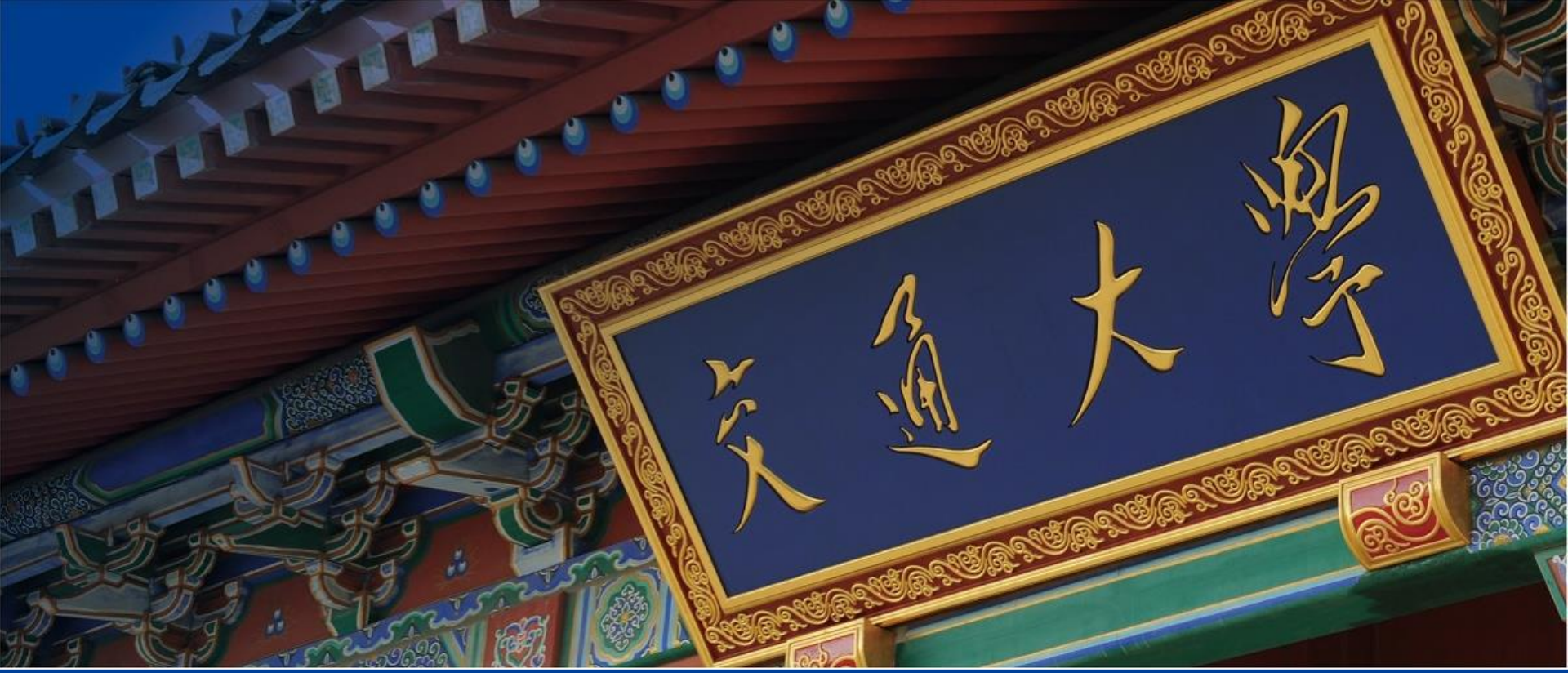

Rectangode: A Reed-Solomon based bar code generator

Xinwei Zhu Ke Chen

2023年5月

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目录 Contents

目录 Contents

Introduction

- **Our work is inspired from the error correction code in Cover's textbook.**
- QR-code used in our daily life is an error correction code as well.
- QR codes consist of black squares arranged on a white background, with data encoded in the pattern of these squares. It holds the ability of error correction by using Reed–Solomon error correction.
- But can QR-code become rectangle?

目录 Contents

Related Work

- **Hamming Code: An easy example**
  - A set of data bits is encoded along with additional parity bits.
  - If the data contains an odd number of 1s, set the parity bit to 1; otherwise, if the data contains an even number of 1s, set the parity bit to 0.
  - Only odd-numbered bit changes can be detected. Cannot correct errors.
- **Hamming Code: Single Error Correcting**

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	✓		✓		✓		✓		✓		✓		✓		✓		✓		✓		
	p2		✓	✓			✓	✓			✓	✓			✓	✓			✓	✓		...
	p4				✓	✓	✓	✓					✓	✓	✓	✓					✓	
	p8								✓	✓	✓	✓	✓	✓	✓	✓						
	p16																✓	✓	✓	✓	✓	

Related Work

- **Hamming Code: Single Error Correcting**
  - Starting from 1, label the data bits with sequential numbers from left to right: 1, 2, 3, 4, 5...
  - Convert the positional numbers of these data bits into binary: 1, 10, 11, 100, 101, and so on.
  - The positions of the data bits that are powers of two (numbers 1, 2, 4, 8, etc., meaning their binary representation has only one 1) are the parity bits.
  - All other positions of data bits (the binary representation of their positional numbers has at least two 1s) are the new data bits.
  - Each data bit is included in specific two or more parity bits.

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	✓		✓		✓		✓		✓		✓		✓		✓		✓		✓		
	p2		✓	✓			✓	✓			✓	✓			✓	✓			✓	✓		...
	p4				✓	✓	✓	✓					✓	✓	✓	✓					✓	
	p8								✓	✓	✓	✓	✓	✓	✓	✓						
	p16																✓	✓	✓	✓	✓	

Related Work

- **Hamming Code: Single Error Correcting**
  - To check for an error in a specific bit, we need to check all the parity bits that include that bit.
  - If all the parity bits are correct, there is no error. However, if any of the parity bits are incorrect, the sum of their positions will identify the erroneous bit.

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	✓		✓		✓		✓		✓		✓		✓		✓		✓		✓		
	p2		✓	✓			✓	✓			✓	✓			✓	✓			✓	✓		...
	p4				✓	✓	✓	✓					✓	✓	✓	✓					✓	
	p8								✓	✓	✓	✓	✓	✓	✓	✓						
	p16																✓	✓	✓	✓	✓	

目录 Contents

Reed-Solomon code:Encoding and Decoding

- In the original study of Reed and Solomon, they construct a code via a polynomial.
- According to Lagrange polynomial, if there are k points, we can generate a polynomial whose indeterminate with the highest degree has a maximal degree of $k - 1$.
- So if we have coefficients $a_0, a_1, \ldots, a_{k-1}$ and variables $x_0, x_1, \ldots, x_{k-1}$, we can encode the coefficients to $f(x_0), f(x_1), \ldots, f(x_{k-1})$, and decoding means getting $a_0, a_1, \ldots, a_{k-1}$ via variables $x_0, x_1, \ldots, x_{k-1}$.

Encoding:

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^{k-1} \\ 1 & x_1 & \cdots & x_1^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{k-1} & \cdots & x_{k-1}^{k-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{k-1}) \end{bmatrix}$$

Decoding:

$$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{bmatrix} = \begin{bmatrix} 1 & x_0 & \cdots & x_0^{k-1} \\ 1 & x_1 & \cdots & x_1^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{k-1} & \cdots & x_{k-1}^{k-1} \end{bmatrix}^{-1} \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{k-1}) \end{bmatrix}$$

Reed-Solomon code: Error Detection

- The number of elements in the Galois field is referred to as its order. It has been proven that the order of Galois field, denoted as q, must be a power of a prime number p, specifically $q = p^n$, where n is a positive integer. Therefore, the Galois field is commonly represented as $GF(p^n)$. The Galois field used in QR codes is $GF(2^8)$.
- We use coefficients as codewords and polynomial values as messages in this case.
- Generate polynomial: $g(x) = (x-\alpha^0)(x-\alpha^1)\cdots(x-\alpha^{n-k-1}) = g_0 + g_1 x + \cdots + g_{n-k-1}x^{n-k-1} + x^{n-k}$
- Multiply first k bits by x^{n-k}. Calculate the remainder using $s(x) = p(x) \cdot x^{n-k} \; mod \; g(x)$. The final polynomial $s(x)$ is equal $to \; p(x) \cdot x^{n-k} - s_r(x)$, coefficients serve as codewords. When the receiver receives $r(x)$, if the code is transferred correctly, we have $r(x) = 0 \; mod \; g(x)$.

Reed-Solomon code: Error Correction

- **Peterson–Gorenstein–Zierler decoder**

- Define sender sends $s(x)$ and receiver receives $r(x)$.
- Define error function $e(x) = r(x) - s(x) = \sum_0^{i-1} e_i x^i$.
- Denote Si assyndrome to ignore the original message, $S_i = r(\alpha_i) = s(\alpha_i) + e(\alpha_i) = 0 + e(\alpha_i) = e(\alpha_i)$, matrix form is on the right.

$$\begin{bmatrix} e_0\alpha^0 + e_1\alpha^0 + \cdots + e_{n-1}\alpha^0 \\ e_0\alpha^{1\times 0} + e_1\alpha^{1\times 1} + \cdots + e_{n-1}\alpha^{1\times(n-1)} \\ \vdots \\ e_0\alpha^{(n-k-1)0} + e_1\alpha^{(n-k-1)1} + \cdots + e_{n-1}\alpha^{(n-k-1)(n-1)} \end{bmatrix} = \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_{n-k-1} \end{bmatrix}$$

$$\begin{bmatrix} \alpha^0 & \alpha^0 & \cdots & \alpha^0 \\ \alpha^{1\times 0} & \alpha^{1\times 1} & \cdots & \alpha^{1\times(n-1)} \\ \vdots & & & \\ \alpha^{(n-k-1)0} & \alpha^{(n-k-1)1} & \cdots & \alpha^{(n-k-1)(n-1)} \end{bmatrix} \begin{bmatrix} e_0 \\ e_1 \\ \vdots \\ e_{n-1} \end{bmatrix} = \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_{n-k-1} \end{bmatrix}$$

Reed-Solomon code: Error Correction

- **Peterson–Gorenstein–Zierler decoder**
- Suppose there are v errors, $X_k = \alpha_{ik}$, $Y_k = e_{ik}$, rewrite the matrix below.

$$\begin{bmatrix} X_0^0 & X_1^0 & \cdots & X_{v-1}^0 \\ X_0^1 & X_1^1 & \cdots & X_{v-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ X_0^{n-k-1} & X_1^{n-k-1} & \cdots & X_{v-1}^{n-k-1} \end{bmatrix} \begin{bmatrix} Y_0 \\ Y_1 \\ \vdots \\ Y_{v-1} \end{bmatrix} = \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_{n-k-1} \end{bmatrix}$$

- Define the error locator polynomial $\Lambda(x) = \prod_{k=1}^{\nu}(1 - X_k x) = 1 + \Lambda_1 x^1 + \Lambda_2 x^2 + \cdots + \Lambda_\nu x^\nu$
- Substituting $x = X_k^{-1}$, $\Lambda(X_k^{-1}) = 0$. Expand it, and multiply $Y_i X_i^{j+v}$ at both sides.

Reed-Solomon code: Error Correction

- **Peterson–Gorenstein–Zierler decoder**
- We can get another inequation:

$$(Y_i X_i^{j+v})\Lambda(X_i^{-1}) = (Y_i X_i^{j+v})(1 + \Lambda_1 X_i^{-1} + \Lambda_2 X_i^{-2} + \cdots + \Lambda_v X_i^{-v}) = 0$$

$$(Y_i X_i^{j+v})\Lambda(X_i^{-1}) = Y_i X_i^{j+v} + \Lambda_1 Y_i X_i^{j+v-1} + \Lambda_2 Y_i X_i^{j+v-2} + \cdots + \Lambda_v Y_i X_i^{j}.$$

- Sum up from $i = 0$ to $i = n - 1$:

$$\begin{aligned}
0 &= \sum_{i=0}^{v-1} Y_i X_i^{j+v} \Lambda(X_i^{-1}) \\
&= \sum_{i=0}^{v-1} \left(Y_i X_i^{j+v} + \Lambda_1 Y_i X_i^{j+v-1} + \Lambda_2 Y_i X_i^{j+v-2} + \cdots + \Lambda_v Y_i X_i^{j}\right) \\
&= \left(\sum_{i=0}^{v-1} Y_i X_i^{j+v}\right) + \Lambda_1 \left(\sum_{i=0}^{v-1} Y_i X_i^{j+v-1}\right) + \Lambda_2 \left(\sum_{i=0}^{v-1} Y_i X_i^{j+v-2}\right) + \cdots + \Lambda_v \left(\sum_{i=0}^{v-1} Y_i X_i^{j}\right) \\
&= S_{j+v} + \Lambda_1 S_{j+v-1} + \Lambda_2 S_{j+v-2} + \cdots + \Lambda_v S_j
\end{aligned}$$

Reed-Solomon code: Error Correction

- **Peterson–Gorenstein–Zierler decoder**
- For every j from $j = 0$ to $j = v - 1$, summarize in matrix form:

$$\begin{bmatrix} S_0 & S_1 & \cdots & S_{v-1} \\ S_1 & S_2 & \cdots & S_v \\ \vdots & \vdots & \ddots & \vdots \\ S_{v-1} & S_v & \cdots & S_{2v-2} \end{bmatrix} \begin{bmatrix} \Lambda_v \\ \Lambda_{v-1} \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_v \\ -S_{v+1} \\ \vdots \\ -S_{2v-1} \end{bmatrix}$$

- Solve $\Lambda(x)$ and let x be α^{-i}, if $\Lambda(x) \neq 0$, it means an error occurred at this position.
- If the position is found, then we can calculate Y_i, thus correcting the answer.

Reed-Solomon code: Property

- We introduce the singleton bound:

Theorem 1 *(Singleton bound)* $A_q(n,d) \leq q^{n-d+1}$.

- Where $A_q(n,d)$ is the maximum number of possible codewords in a q-ary block code of length n and minimum distance d.
- In Reed-Solomon code, the maximum number of possible codewords is q^k, so $q^k \leq q^{n-d+1}, k \leq n - d + 1$.
- The minimum hamming distance of Reed-Solomon code is $n - d + 1$, so we can detect up to $n - k$ errors and correct $\lfloor \frac{n-d}{2} \rfloor$ errors.

目录 Contents

Rectangode: Constructing

- The Rectangode consist with three parts, including the position symbols, the format information, and the data bits.
  - The position symbols help to identify the Rectangode and locate the data. There are two symbols, one on the left-up side of the Rectangode, and the other on the right-down side of the Rectangode.
  - The format information tells us the length of the code and the length of the additional bits for correction. It's on the right side of the left-up symbol.
  - The data bits are the Reed-Solomon code of the original codeword. It has a flexible code length to adjust to different types of demand.

Rectangode: Implementation

- The source code of this part is on https://github.com/Gazer2020/Rectangode/, composed of the encoding part, the bit2image part, the image2bit part, and the decoding part.
  - 1. The encoding part utilizes reedsolo to help converting string to bits, using Reed-Solomon code by a flexible k to change the error-correcting code length.
  - 2. The bit2image part converting the bits to numpy 2-D array, adding some other information, such as the position symbol and an 8-bit number indicating the length of the error-correcting code. Then the matrix is transformed into an image by the PIL library.
  - 3. The image2bit part converting the image back to numpy 2-D array.
  - 4. The decoding part is using reedsolo to convert bit to readable strings, with error correction on it.

Rectangode: Example

- (Encoding “My name’s Chen Ke. I’m from SJTU.”)
- Basic Ability: Figure 1
- Error Correction: Figure 2
- Flexibility: Figure 3(Parameters set as Table 1)

row	30
col	20
k	28

Table 1: parameter setting

Figure 1: basic

Figure 2: error

Figure 3: flexibility

目录 Contents

Conclusion

- RS codes can handle burst errors, consecutive errors that occur during transmission or storage.
- By utilizing Reed-Solomon codes, new barcode designs with flexible shapes can be created, enhancing their aesthetic appeal and meeting various application requirements.

Thanks:

- **Thanks to Prof. Fan Cheng and course assistants for your dedication to this course!**

Thanks!