Rectangode: A Reed-Solomon based bar code generator

Zhu Xinwei, Chen Ke

May 25, 2023

Abstract

This essay explores the use of Reed-Solomon codes in barcode generation, specifically focusing on the development of a new type of 2-D shaped barcode that can adapt to various contours and artistic expressions. The motivation for utilizing Reed-Solomon codes stems from the need for more flexible and visually appealing barcode designs that can go beyond traditional rectangular or square shapes. The essay begins with an overview of QR codes, highlighting their popularity and functionality. It then delves into the principles of Reed-Solomon codes, discussing their encoding and decoding processes, as well as error detection and correction techniques. The Peterson-Gorenstein-Zierler decoder for Reed-Solomon codes is introduced as a method for error correction. The essay concludes by showcasing the potential of Reed-Solomon codes in creating unique 2-D shaped barcodes that can cater to different applications, such as advertisements and art installations.

1 Introduction

1.1 QRcode:An overview

Quick Response (QR) codes have gained significant popularity in recent years due to their ability to store large amounts of information in a compact, two-dimensional format. QR codes consist of black squares arranged on a white background, with

data encoded in the pattern of these squares. These codes are commonly scanned using smartphones or dedicated barcode scanners, allowing users to access information quickly by simply capturing an image.

1.2 Motivation for Reed-Solomon Codes in Barcode Generation

While QR codes have proven to be a versatile tool, there is a growing need for more flexible and visually appealing barcode designs. Traditional barcode formats are limited to rectangular or square shapes, which may not always align with the aesthetics or requirements of different applications, such as advertisements and art installations. To address this limitation, we propose the utilization of Reed-Solomon codes to develop a new type of 2-D shaped barcode that can adapt to various contours and artistic expressions. We first introduce the principle of Reed-Solomon code and its encoder and decoder, then we try to generate the Rectangode, a special 2D barcode which has a flexible shape.

2 Related work

Before the invention of Reed-Solomon code, there exists some simple error-correction code, such as hamming code. [1]

2.1 Construction of Hamming Code

To construct a Hamming code, a set of data bits is encoded along with additional parity bits. The number and position of these parity bits are determined by the desired error-correction capabilities. The codeword is structured in such a way that each bit within the codeword participates in multiple parity calculations, allowing for the detection and correction of errors.

2.2 Error Detection and Correction

During the decoding process, the receiver checks the received codeword for errors using the calculated parity bits. If an error is detected, the receiver utilizes the parity information to identify the erroneous bit. By flipping the erroneous bit, the receiver can correct the single-bit error. The Hamming code can detect and correct single-bit errors, but it cannot correct multiple-bit errors.

3 Reed–Solomon code

Reed-Solomon (RS) codes are a type of error-correcting codes that are widely used in various communication systems and data storage applications. They provide a way to add redundancy to transmitted or stored data, allowing for the detection and correction of errors that may occur during transmission or storage.

The principle behind Reed-Solomon codes involves representing the data to be transmitted or stored as a sequence of symbols. These symbols are typically represented by numbers, and their length is referred to as the code length. The code length determines the number of symbols in each codeword.

Reed-Solomon codes are designed to handle burst errors, which are consecutive errors that may occur in a communication channel or during storage. The codes use a technique known as polynomial division to generate redundant symbols, which are appended to the original data to create a codeword. These redundant symbols contain information that allows the receiver to detect and correct errors. We will discuss it later.

3.1 Encoding and Decoding

In the original study of Reed and Solomon [2], they construct a code via polynomial. According to Lagrange polynomial, if there are k points, we can generate a polynomial whose indeterminate with 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 $x_0, x_1, \ldots, x_{k-1}$ and $f(x_0), f(x_1), \ldots, f(x_{k-1})$. The message m is equal

to $a_0a_1 \dots a_{k-1}$. The codeword C is equal to $f(x_0)f(x_1) \dots f(x_{k-1})$. Rewrite the whole process in matrix form:

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:

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

Upon observation, the square matrix is a Vandermonde matrix, with a determinant of $\prod_{0 \le i < j \le k}(x_j - x_i)$. As each value of x is unique, this expression is always greater than 0, allowing for its inverse.

There is an alternative method of encoding that utilizes coefficients as codewords and polynomial values as messages. The process is similar, so we will not repeat it here.

3.2 Error Detection

To introduce this method, we first introduce the finite field, also known as the Galois field [3]. 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. For a codeword of length n, we aim to have the first k bits represent our message, while the remaining $n - k$ bits serve as error-correction bits.

We define our generator polynomial as $g(x)$, which has roots as powers of α. Here,

α represents the n-th unit root, such that:

$$g(x) = \left(x - \alpha^0\right)\left(x - \alpha^1\right)\cdots\left(x - \alpha^{n-k-1}\right) = g_0 + g_1 x + \cdots + g_{n-k-1}x^{n-k-1} + x^{n-k}$$

Once we have generated the first k bits using $f(x)$, we multiply it by x^{n-k}. Then, we calculate the remainder using $s_r(x) = p(x) \cdot x^{n-k} \mod g(x)$. The final polynomial $s(x)$ is equal to $p(x) \cdot x^{n-k} - s_r(x)$, and its coefficients serve as codewords. When the receiver receives $r(x)$, if the code is transferred correctly, we have $r(x) = 0 \mod g(x)$. Otherwise, an error exists.

3.3 Error Correction

Reed-Solomon code offers ability of error correction. There are many decoders for Reed-Solomon code, and We will only introduce the Peterson–Gorenstein–Zierler decoder. [4]

First we need to locate where the error is. To locate error, we do some preparations. Suppose $s(x)$ is what the sender send and $r(x)$ is what the receiver received. We define $e(x) = \sum_{i=0}^{n-1} e_i x^i$ as the difference between $r(x)$ and $s(x)$, that is, $e(x) = r(x) - s(x)$. Since $s(x) = 0 \mod g(x)$, $r(x) = e(x) \mod g(x)$. We denote S_i as syndrome to ignore the original message, $S_i = r(\alpha_i) = s(\alpha_i) + e(\alpha_i) = 0 + e(\alpha_i) = e(\alpha_i)$, where $0 \le i < n - k$. Rewrite it in matrix form:

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

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

For the convenience of further discussion, we suppose there are v errors, and $X_k = \alpha^{i_k},\ Y_k = e_{i_k}$.The matrix form is as below:

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

Then we define the error locator polynomial

$$\Lambda(x) = \prod_{k=1}^{\nu}(1 - X_k x) = 1 + \Lambda_1 x^1 + \Lambda_2 x^2 + \cdots + \Lambda_\nu x^\nu$$

By substituting $x = X_k^{-1}$, $\Lambda(X_k^{-1}) = 0$. Expand it, and multiply $Y_i X_i^{j+v}$ at both sides. We can get another equation:

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

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

Sum up from $i = 0$ to $i = v - 1$.

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

For every j from 0 to $v - 1$, summarize in matrix form:

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

Then we can 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.

3.4 Property

We first introduce Singleton bound. Define the minimum distance of a set C of codewords of length n as d, $d = \min_{\{x,y \in C: x \neq y\}} d(x,y)$, where $d(x,y)$ is the hamming distance of x and y. Denote $A_q(n,d)$ as the maximum number of possible codewords in a q-ary block code of length n and minimum distance d.Then we have:

Theorem 3.1 *(Singleton bound)* $A_q(n,d) \leq q^{n-d+1}$.

In Reed-Solomon code, the maximum number of possible codewords is q^k, so $q^k \leq q^{n-d+1}$, $k \leq n-d+1$.

According to the inequality, the minimum Hamming distance of Reed-Solomon code is given by $d_{min} = n-k+1$. This implies that we can detect up to $n-k$ errors, and if we are unaware of the error positions, we can correct $\lfloor \frac{n-k}{2} \rfloor$ errors.

The rate of Reed-Solomon code is $R = \frac{k}{n}$, when $k = n$ achieves maximum rate. However, when in real use, we often choose $R = \frac{1}{2}$.

4 Rectangode

4.1 Background

In our tangible reality, QR codes have already permeated society at large, propagating extensively. The act of scanning a QR code and accessing relevant information has become a remarkable convenience. However, QR codes still possess certain drawbacks. One such limitation lies in their inherent square shape, which may prove challenging when attempting to affix them onto various surfaces with diverse contours. In light of this quandary, we propose the construction of a novel breed of two-dimensional barcodes, featuring a rectangular structure, thereby rendering them adaptable to a wider array of scenarios and circumstances.

4.2 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.

An example of the Rectangode is

4.3 Implementation

We create a python implementation of our Rectangode on Rectangode. By some open source python library, including numpy, reedsolo and PIL, we build the project. The structure of the project are four parts, 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.

4.4 Test

To verify the project, we do three tests. The first test shows the fundamental ability of the project. The second test shows the error correcting ability of Rectangode. The third test shows the flexibility of the shape.

4.4.1 Basic ability

We set the string to be "My name's Chen Ke. I'm from SJTU." Then the output image is Figure 1.

Figure 1: basic

Figure 2: error

Figure 3: flexibility

4.4.2 Error correction

We modified Figure 1 by make its right-down angle all white. Then we read the image and still get the right result.

4.4.3 Flexibility

We can customize the shape of Rectangode and the length of the error correction code. Figure 3 is an example. The parameters are Table 1.

row	30
col	20
k	28

Table 1: parameter setting

5 Conclusion

QR codes have become popular due to their convenience to transmit information in a compact, two-dimensional format. However, there is a need for more flexible and visually appealing barcode designs. To address this, the utilization of Reed-Solomon codes is proposed to create a new type of 2D barcode that can adapt to various contours and artistic expressions. Reed-Solomon (RS) codes are error-correcting codes widely used in communication systems and data storage. They add

redundancy to data, allowing for error detection and correction. RS codes represent data as symbols and use polynomial division to generate redundant symbols, which are appended to the original data. The receiver checks for errors using calculated parity bits and can correct single-bit errors. 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.

References

[1] Richard W Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[2] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[3] Harold M Edwards and Harold M Edwards. *Galois theory*, volume 19840. Springer New York, 1984.

[4] Daniel Gorenstein and Neal Zierler. A class of error-correcting codes in p^m symbols. *Journal of the Society for Industrial and Applied Mathematics*, 9(2):207–214, 1961.