Reed-Solomon 알고리즘을 이용한 2차원 바코드 시스템에서 오류 복구 기능 설계 및 구현

장 승 주†

요 약

본 논문은 2차원 바코드 시스템에서 데이터 오류에 대한 복구 기법을 설계 및 구현하였다. 데이터 오류 복구 기법으로 Reed-Solomon 알고리즘을 수정하여 이용하였고 2차원 바코드 시스템을 실제 설계 및 개발하였다 실제 개발된 2차원 바코드 시스템에 대한 내용을 언급하고 오류 복구를 위한 ECC(Error Correction Code)를 직접 설계 및 구현하였다. 2차원 바코드 시스템에 구현된 ECC코드에 대한 구체적인 설명을 한다 2차원 바코드 시스템은 데이터 저장 용량을 달리하는 7가지 타입을 갖는다. 7가지 2차원 바코드 타입에 따라 ECC코드를 구현했을 경우 여러 가지 실험을 했다. 실험 내용은 각 2차원 바코드 타입에 따른 ECC코드와 데이터 저장 용량의 비율을 산출한다 이 실험 결과 데이터 저장이 많이 되는 타입일수록 실제 데이터 저장 용량은 적어진다는 것이다 이것은 ECC 코드가 차지하는 비중이 늘어남을 알 수 있다. 그리고 오류 복구율 실험에서 2차원 바코드 시스템이 35~40% 정도의 오류에 대해서는 복구가 가능함을 확인했다.

Design and Implementation of Reed-Solomon Code for 2-Dimensional Bar Code System

Seung-Ju Jang†

ABSTRACT

This paper is designed and implemented the data recovery mechanism for 2-D (2-dimensional) bar code system. The data recovery algorithm used the modified Reed-Solomon algorithm and it is implemented into 2-D bar code system. There are 7 types of 2-D bar code system, which are 21x21, 25x25, 41x41, 45x45, 73x73, 101x101, 177x177 This paper has been experimented that how many data is saved among several 2-D bar code types and how many data are recovered. In the first experiment, the big size 2-D bar code system has many ECC codeword. Therefore, original data cannot be assigned to 2-D bar code system In the second experiment, even if 35~40% loss data for the 2-D bar code system, the 2-D bar code system could have been recovered to original data.

1. 서 론

컴퓨터와 통신의 발달로 정보화시대가 도래하면서 데이터의 수집 및 이용방법이 개인 및 기업, 사회전반에 걸쳐 매우 중요한 부분으로 대두되고 있다. 하지만 아직도 사회 전반적으로 수작업에 의한 방법을 탈피하지 못하여 인적, 물적 손실을 초래하고 있는 현 시점에서 바코드의 이용은 주먹구구식의 비능률적이고 부정확한 기존의 사무처리를 효과적이고 신속, 정확하게 처리할 수 있는 새로운 정보의 관리 및 체계화를 위해서 필요한 기술이다. 현재 전 세계적으로도 널리 이용되고 있는 바코드는 국내에서도 그 효용가치가 인정되면서 유통업을 중심으로 사무자동화, 공장 자동화등

※ 본 논문은 1999년도 동의대학교 학술연구조성비에 의해 연구되었음.
† 정 회 원 : 동의대학교 컴퓨터공학과 교수
논문접수 : 1999년 11월 6일, 심사완료 : 2000년 5월 10일

다양한 업무분야에 이용되고 있다. 그러나 현재 사용 및 유통되고 있는 바코드 시스템은 1차원 바코드 시스템이다.

1차원 바코드 시스템은 여러 가지 문제점을 가지고 있다. 1차원 바코드 자체가 데이터 정보를 가지는데 한계가 있다는 것과 바코드 자체가 유실되었을 경우 아무런 대책이 없다는 것이다

2차원 바코드에 대한 연구는 AIM Internation, Inc를 중심으로 활발히 연구가 이루어지고 있다. 1차원 바코드를 대체할 차세대 바코드로 2차원 바코드에 대한 연구가 활발히 이루어지고 있다. AIMI에서 발표된 2차원 바코드의 기본적인 구조는 실제 데이터를 정확히 저장할 수 있는 능력을 갖는다는 것, 데이터의 양에 따라 다양한 모델을 지원해 준다는 것. error가 발생할 경우에 대한 복구율등이 명시되어 있다는 것, 코드의 모양은 matrix 형태라는 것 등이다[1-3]. 현재 일반적으로 사용되고 있는 2차원 바코드는 니혼덴소의 QR코드, 데이터 코드등이 있다[4].

따라서 최근에 1차원 바코드를 대체할 2차원 바코드에 대한 연구가 활발히 진행되고 있다. 2차원 바코드 시스템은 데이터를 자체적으로 가질 수 있다. 따라서 컴퓨터 시스템의 본체와 연결시키는 번거로운 작업이 필요없다. 자신에게 필요한 정보를 2차원 바코드 자체에 모두 표현할 수 있기 때문이다. 그리고 바코드의 유실로 인한 중요한 정보가 손실될 경우 자체 복원 기능을 갖도록 하기 위한 연구가 진행되고 있다[4, 5].

일반적으로 ECC(Error Correction Code) 알고리즘은 1948년 Claude Shannon에 의하여 세상에 발표되었다. Shannon이 제안한 ECC는 채널이라는 개념으로 연결될 수 있음을 보였다. 그러나 구체적으로 적당한 코드를 제시하지 못했다. 1960년대에 이러한 대안으로 블록 코드(block code)가 출현하게 되었고 최초의 블록 코드가 해밍 코드(Hamming code)이다[6, 12]. 이후에 많은 발전을 거듭한 끝에 multiple-error correcting code로 BCH코드, Reed-Solomon code가 개발되게 되었다[6, 8, 10, 11]. 이러한 코드를 기반으로 하여 통계학적인 접근을 시도하게 되고 기존코드의 표현 방식을 트리(tree) 형식으로 하게 된다. 이러한 계층적인 코드를 convolution code라고 한다. 나중에 앞의 두가지 방법들을 통합한 코드를 개발하게 된다. 현재 디지털 환경인 컴퓨터에 가장 적절히 표현될 수 있는 코드로 Reed-Solomon code를 사용하고 있다. 특히 컴퓨터 하드웨어나 소프트웨어에서 오류를 허용하지 않는 환경에서 ECC 코드의 사용은 필수적이다[7, 11].

2차원 바코드 시스템에 대한 국제 표준화 활동은 AIMI를 중심으로 진행되고 있다. 그리고 2차원 바코드에서 ECC코드의 적용은 각 개별 단체를 중심으로 진행되고 있다. 대부분의 경우 BCH 코드나 변형된 Reed-Solomon 코드를 사용하고 있다[4]. 본 논문은 2차원 바코드 시스템에서 ECC코드를 개발한다. 2차원 바코드 시스템은 그 자체에 데이터 저장 능력을 갖는다. 따라서 2차원 바코드 자체 구조체의 손상으로 인한 데이터 손실이 발생할 가능성이 높다. 손실된 데이터에 대한 복구 능력을 갖지 않을 경우 치명적인 결과를 초래할 수 있다. 따라서 본 논문은 2차원 바코드 시스템을 개발하고 개발된 2차원 바코드 시스템에 변형된 Reed-Solomon 알고리즘을 사용하여 실제 구현을 한다. 실제 구현된 2차원 바코드 시스템을 이용하여 실험을 한다. 실험 내용은 2차원 바코드 시스템에 ECC코드를 적용하지 않은 경우의 데이터 저장 용량과 ECC 코드를 적용했을 경우 저장할 수 있는 용량의 비율을 측정한다. 그리고 개발된 ECC 코드를 갖는 2차원 바코드 시스템에서 오류 복구율을 실험했다.

본 논문의 구성은 2장에서 2차원 바코드 시스템에 대한 설명, 3장에서는 2차원 바코드 시스템에서 Reed-Solomon 알고리즘을 구현한 내용, 4장에서는 실험한 내용, 5장 결론의 순으로 언급한다.

2. 2차원 바코드 시스템

2차원 바코드 시스템은 기존의 바코드(1차원 바코드) 시스템에 비하여 유연성과 기억 용량이라는 측면에서 엄청난 차이를 보인다. 1차원 바코드의 경우는 정해진 형식에 따라 정해진 데이터가 한정된 크기로 저장된다. 2차원 바코드는 데이터 정보를 관리하기 위한 헤더 정보를 제외한 나머지 데이터 부분은 유연성 있게 저장이 가능하다

1차원 바코드 시스템은 데이터 저장 형태가 1차원 평면에 저장된다. 따라서 데이터 형식이 전세계적으로 표준화되어 있다. 반면 2차원 바코드 시스템은 데이터 저장 형태가 2차원 평면에 저장된다. 2차원 평면에 저장되기 때문에 많은 자료를 저장가능하다.

본 논문에서 개발한 2차원 바코드 시스템은 여러 가지 모듈로 구성된다. 이러한 2차원 바코드 시스템 모

듈은 다음과 같다.

- 2차원 바코드 시스템 인터페이스 모듈
- 2차원 바코드 시스템 encoding 기능 모듈
- 2차원 바코드 시스템 decoding 기능 모듈

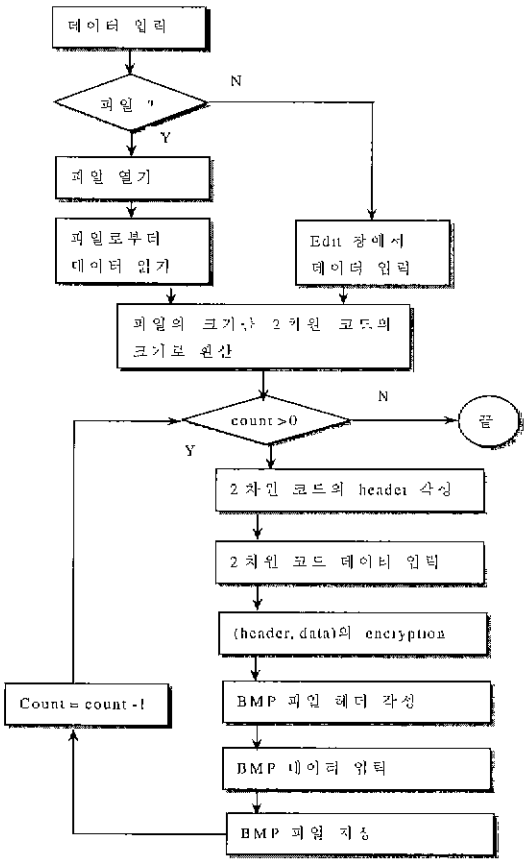
2.1 2차원 바코드 시스템 사용자 인터페이스

2차원 바코드 시스템 사용자 인터페이스 모듈은 2차원 바코드 시스템 기능을 사용자들이 편리하게 사용할 수 있는 환경을 제공한다. 사용자 인터페이스 환경은 시각적인 기능의 제공으로 사용자들이 편리하게 사용할 수 있도록 한다. 2차원 바코드는 데이터의 양에 따라 여러 가지 타입을 갖는다. 2차원 바코드 타입은 21x21, 25x25, 41x41, 45x45, 73x73, 101x101, 177x177 7가지가 있다. 사용자 인터페이스 모듈은 이러한 타입을 사용자들이 선택할 수 있도록 해준다.

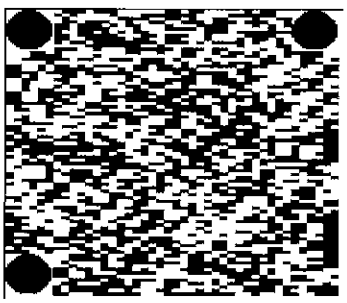
2.2 2차원 바코드 시스템 인코딩(encoding) 모듈

2차원 바코드 시스템 인코딩 모듈은 사용자가 입력한 데이터를 2차원 바코드로 변환하는 모듈이다. 인코딩 모듈은 모든 형식(txt, doc, hwp, gif, bmp, exe등)의 파일에 대해서 변환이 가능한 특징을 갖는다. 2차원 바코드 시스템에서 인코딩 모듈의 수행 흐름은 다음 (그림 1)과 같다.

2차원 바코드를 생성하기 위하여 먼저 데이터를 사용자로부터 입력받아야 한다. 사용자로부터 입력되는 데이터는 입력창으로부터 직접 또는 파일로부터 받는 경우의 두 가지가 있다. 입력창에서 데이터를 입력 받을 경우는 윈도우 버퍼를 통해서 데이터를 직접 받으면 된다 파일일 경우는 파일로부터 데이터 입력을 받기 위하여 먼저 파일 열기(file open)를 수행한다. 그런 후 파일로부터 데이터를 입력받는다. 파일로부터 읽어들인 데이터는 2차원 코드 하나에 들어갈 수 없는 경우에 만들어야 하는 2차원 코드 개수를 결정한다. 만들어야 하는 2차원 코드의 갯수 만큼 다음의 과정을 반복 수행한다. 2차원 코드 작성을 위한 헤더를 만든다. 이 헤더에 들어가는 정보는 파일 이름, 파일 크기, 비밀 번호, 키 값등이 포함된다. 그리고 나서, 2차원 코드 작성을 위한 데이터를 입력한다. 데이터 입력이 완료되면 2차원 코드에 들어있는 모든 데이터에 대한 암호를 수행한다 암호 수행이 정상적으로 수행되면 이 데이터를 이미지 파일로 변환하는 작업을 수행한다. 이미지 파일(BMP 파일)로 변환을 하기 위하여 BMP 헤더를 작성하고, BMP 데이터를 입력한다. 이 작업이 끝나면 BMP 파일 형식으로 저장을 하게 된다. 이 과정을 수행하고 나면 다음 (그림 2)와 같은 모양의 2차원 코드가 생성된다.



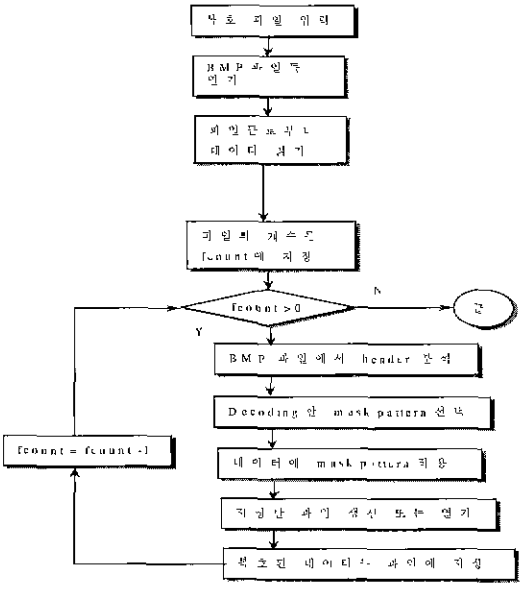
(그림 1) 2차원 바코드 시스템에서 데이터 encoding을 위한 모듈 수행 흐름도



(그림 2) 2차원 바코드의 모양

2.3 2차원 바코드 시스템 디코딩(decoding) 모듈

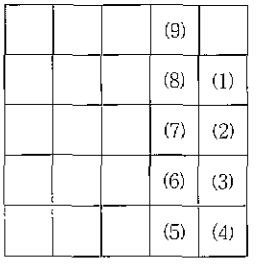
2차원 바코드 디코딩(decoding) 모듈은 2차원 바코드로 만들어진 것을 일반문서 형식으로 복원해 내는 부분이다. 디코딩 모듈은 2차원 바코드의 헤더에 여러 가지 정보를 갖는다. 이 헤더 정보에서 디코딩 결과 만들어야 하는 파일 이름과 형식을 추출한다. 추출된 파일 이름과 형식에 대해서 디코딩 내용을 저장한다. 파일 형식(txt, doc, hwp, exe, jpg, bmp를 포함한 모든 파일 형식)은 어떠한 파일이라도 가능할 수 있도록 하고 있다.



(그림 3) 2차원 바코드 시스템에서 데이터 디코딩을 위한 모듈 수행 흐름도

2차원 바코드로 만들어진 것을 원래 데이터로 복원하는 기능을 하는 것이 데이터 디코딩 수행 모듈이다. 데이터를 복호하기 위하여 먼저 2차원 바코드가 저장된 여러개의 BMP 파일 중 하나를 지정한다. 그러면 이것과 연관된 파일 중 첫 번째 파일에 대한 복호 과정을 수행한다. 먼저 복호하고자 하는 이미지 파일의 개수를 fcount에 저장한다. fcount가 0이 아닐 때까지 다음의 과정을 반복 수행한다. BMP 파일을 연다(file open). 파일로부터 데이터를 버퍼에 읽어들인다. 읽어들인 파일에서 헤더 정보를 분석한다(파일 이름, 패스워드, 파일 크기, 파일 형식등). 데이터에 대해서 decoding 할 알고리즘을 적용한다. 원래의 파일로 저장하기 위하여 저장할 파일의 생성 또는 열기를 수행한다(file creation or file open). 복호된 데이타를 파일에 저장한다. 개발에 사용된 프로그래밍 환경은 VC++ 언어를 이용하였다.

구현된 2차원 바코드 시스템은 정사각형 행렬(matrix) 모양을 하고 있다. 가로와 세로의 길이가 동일하며 데이터가 들어가는 공간도 일정하다. 2차원 바코드의 모양은 다음과 같다. 2차원 바코드의 모서리에 동근 모양은 2차원 바코드의 position detector이다. position detector는 2차원 바코드에서 데이터의 시작점이 어디인지를 판별하는 기능을 하는 역할을 수행한다. 본 논문에서 개발된 2차원 바코드에서 데이터 입력 형태는 다음 (그림 4)와 같다.



(그림 4) 2차원 바코드에 저장되는 데이터의 모양

2차원 바코드에 저장되는 데이터는 (그림 4)와 같이 오른쪽 하단에서부터 입력된다. 데이터가 입력되는 순서는 (그림 4)에서 번호를 매긴 순서대로 이루어진다. 그리고 앞부분에 삽입되는 정보는 2차원 바코드를 유지하는데 필요한 헤더 정보들이다. 헤더 정보는 데이터의 크기 정보, 사용자에 대한 비밀번호, 여러 개의 2차원 바코드로 구성되어 있을 경우에 이에 대한 정보 등이 들어있다. 위의 그림에서 음영처리된 부분은 바코드 스캐너가 데이터 인식을 할 경우에 위치 정보로 활용한다. 위치 정보를 이용하여 데이터의 시작점을 찾는 것이다

3. 2차원 바코드 시스템에서의 Reed Solomon 알고리즘 구현

2차원 바코드 시스템은 1차원 바코드 시스템이 갖는 한계를 극복하기 위하여 개발되었다. 1차원 바코드 시스템은 13자리로 표현되는 인덱스 방식을 사용하고 있다. 따라서 이 인덱스에 해당하는 데이터 정보를 갖는

시스템이 있어야 한다. 이러한 1차원 바코드 시스템의 문제는 2차원 바코드 시스템에 대한 연구를 촉발하게 하였다. 또한 1차원 바코드 시스템은 바코드 자체가 유실될 경우에 오류 조정 및 검증 기능이 있다

2차원 바코드 시스템은 1차원 바코드 시스템이 갖는 이와같은 문제를 해결하기 위하여 대두되게 되었다. 2차원 바코드 시스템은 바코드 자체가 데이터를 갖는 저장 기능이 있다. 데이터 저장 기능을 갖는다는 것은 1차원 바코드처럼 자료를 보관하는 시스템이 필요없다는 것이다. 그리고 2차원 바코드의 특징은 ECC 기능을 가질 수 있다는 것이다. ECC 기능을 통해서 바코드 자체가 일부 유실될 경우 이것을 보증하는 기능을 담당한다. 본 논문에서는 ECC 코드로 Reed-Solomon 알고리즘을 사용하였다.

3.1 Reed-Solomon 알고리즘 구현

심볼 알파벳(symbol alphabet)을 여러 개의 블록 길이(blocklength)로 나눈다. m = 1이고 심볼 필드(symbol field) GF(q)와 오류 위치 필드(error-locator field) GF(qm)는 같은 값을 갖는다. 일반적으로 n = qm - 1 = q - 1이다. 원소(element) b의 GF(q)에서 최소 다항식(minimal polynomial) 은 fb(q) = x - b이다. 왜냐하면 심볼 필드(symbol field)와 오류 위치 필드는 같은 필드이기 때문이고 모든 최소 다항식은 선형(linear)이다. t-오류 수정(error-correcting) Reed-Solomon code에 대한 (j0 = 1) 생성 다항식(generator polynormal) g(x) = (x-a)(x- a2)...(x-a2t)이다. 이 함수는 2t차수의 다항식이다. 따라서 Reed-Solomon code는 n-k = 2t이다.

즉, GF(q), (where q > 2)에 대한 Reed-Solomon code는 다음과 같다. a를 GF(a)내의 근원적인 원소(primitive element)로 두고, $a_{n-1} = 1$ ai ! = 1(for i! = 0 mod q-1)로 둘 경우 n = q-1 길이(length)와 차수 k인 Reed-Solomon code는 다음 식 (1)의 다항식으로 산출가능한 사이클 코드(cyclic code)이다.

$$g(x) = (x-a)(x-a_2) \ldots (x-a_{n-k-1})(x-a_{n-k}) \quad (1)$$

왜냐하면 각 ai는 루트 유닛(root unity)이고 x-ai divides x_{n-1}이다. 그래서 g divide x_{n-1}이고 사이클 코드이다.

똑같은 방법으로 차수가 <= n-1이고 root로 a, a_2, ..., a_{n-k}인 GF(q), 즉, F가 코드이고 deg(F) <= n-1이고 F(a) = F(a2) = ... = F(a_{n-k}) = 0인 Reed-Solomon code는 다음과 같다. 이러한 성질을 이용하여 Reed-Solomon code에 대한 패리티 체크 메트릭스(parity check matrix)를 찾을 수 있다. F(x) = F_0 + F_1x + .. + $F_{n-1}x_{n-1}$를 코드로 가정한다. 1 <= I <= n-k인 경우

$$F(a_I) = F_0 + F_1a_I + \ldots + F_{n-1}a_I(n-1) = 0 \quad (2)$$

즉, 위의 (2) 다항식은 codeword(F_0, F_1, .., F_{n-1})는 vectors(1, a_1, a_{2i}, ..., $a_I(n-1)$) (1 <= I <= n-k)에 대해서 orthogonal이다. 이러한 vector는 RS code에 대한 parity check matrix의 row에 해당된다 GF(q)의 [n, k] RS code의 패리티 체크 메트릭스는

$$H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdot & \cdot & \cdot & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdot & \cdot & \cdot & \alpha^{2(n-1)} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & \alpha^{n-k} & \alpha^{(n-k)2} & \cdot & \cdot & \cdot & \alpha^{(n-k)(n-1)} \end{pmatrix}$$

[Theorem] Reed-Solomon code는 최소-거리 코드(maximum-distance code)이고 최소 거리는 n-k-1이다

(증명)

d = 2t+1로 둔다. 최소 거리 d는 d* >= d = 2t+1 = n-k+1이다. 왜냐하면 Reed-Solomon에 대해서 2t = n-k이다. 그러나 선형 코드(linear code)에 대한 Singleton bound에 의하여 d <= n-k+1이다. 따라서 d* = n-k+1이고 d* = d이다.

3.2 Decoding of RS codes : 핵심적인 방정식

C를 [n, k] RS code로 나타내고 C에서 코드워드(codeword) $F(x) = \sum_{i=0}^{n-1} F_i x^i$가 송신되고, $R(x) = \sum_{i=0}^{n-1} R_i x^i$가 수신된다. F와 R는 오류 벡터(Error vector) $E(x) = \sum_{i=0}^{n-1} E_i x^i$와 연관된다. 여기서 $R(x) = F(x) + E(x)$이다 디코드(Decoder)는 $E(x)$를 찾기 위한 시도를 한다. $i \le j \le n-k$에 대하여 식 (3)을 얻는다.

$$S_i = R(\alpha^i) = \sum_{i=0}^{n-1} R_i \alpha^{ij} = \sum_{i=0}^{n-1} E_i \alpha^{ij} \quad (3)$$

오류 위치의 {0, 1, .., n-1)의 부분집합을 ε으로 둔다. 즉 $\varepsilon = \{l : E_l \neq 0\}$. 따라서 식 (3)은 다음과 같이 나타낼 수 있다.

$$S_j = \sum_{i \in \varepsilon} E_i \alpha^{ij}, \quad 1 \le j \le n-k \tag{4}$$

디코드는 오류 집합(Error set) ε과 오류값 E_i를 찾는다. 만약 오류의 수가 s이고 $2s \le n-k$이면 식 (4)에 의하여 일정한 해를 얻는다. 그러나 이 해는 비선형적(non-linear)이기 때문에 직접 얻기는 매우 어렵다.

[n, n-2] 1-byte correcting RS code의 경우에 정확히 1개의 Error가 발생될 경우 S_1과 S_2를 얻는다. 이를 위치 i에 있다고 말하고 식 (4)에 의하여 식 (5)을 얻는다.

$$S_1 = E_i \alpha^i \text{ and } S_2 = E_i \alpha^{2i} \tag{5}$$

여기서 $\alpha^i = S_2/S_1$이고 error 위치 i를 결정할 수 있다. Error Value $E_i = (S_1)^2/S_2$이다.

[보기]

GF(8)인 [7, 5, 3] RS code를 고려해보자. 수신 벡터 $x = (101\ 001\ 110\ 001\ 011\ 010\ 100)$를 디코드하고자 한다. 이를 다항식으로 나타내면

$$R(x) = \alpha^6 + \alpha^2 x + \alpha^3 x^2 + \alpha^2 x^3 + \alpha^4 x^4 + \alpha x^5 + x^6$$

$S_1 = R(\alpha) = \alpha^2$ 그리고 $S_2 = R(\alpha^2) = \alpha^4$을 얻는다. 따라서 $S_2/S_1 = \alpha^2$이다. 이는 의치 2에 오류가 있음을 나타낸다. 오류값 $E_2 = (S_1)^2/S_2 = (\alpha^2)^2/\alpha^4 = 1$이고 벡터 형태는 100이다. 디코드의 결과는

$$c = (101\ 001\ 010\ 001\ 011\ 010\ 100)$$

다항식의 형태로 나타내면

$$R(x) = \alpha^6 + \alpha^2 x + \alpha x^2 + \alpha^2 x^3 + \alpha^4 x^4 + \alpha x^5 + x^6$$

차수가 $\le n-k-1$이고 계수가 n-k syndromes인 syndrome 다항식을 식 (6)과 같이 정의하자

$$\begin{aligned} S(x) &= S_1 + S_2 x + S_3 x^2 + \ldots + S_{n-k} x^{n-k-1} \\ &= \sum_{j=0}^{n-k-1} S_{j+1} x^j \end{aligned} \tag{6}$$

$S(x) = 0$이면 R(x)는 C에 있음을 주의하자.

$$S(x) = \sum_{j=0}^{n-k-1} S_{j+1} x^j$$

$$\begin{aligned} &= \sum_{j=0}^{n-k-1} \left(\sum_{i=0}^{n-1} R_i \alpha^{i(j+1)} \right) x^j \\ &= \sum_{i=0}^{n-1} R_i \alpha^i \sum_{j=0}^{n-k-1} (\alpha^i x)^j \\ &= \sum_{i=0}^{n-1} R_i \alpha^i \frac{(\alpha^i x)^{n-k} - 1}{\alpha^i x - 1} \\ &= \sum_{i=0}^{n-1} R_i \frac{(\alpha^i x)^{n-k} - 1}{x - \alpha^{-i}} \end{aligned}$$

$a \ne 1$에 대하여 $\sum_{i=0}^{m} a^i = (a^{m+1} - 1)/(a-1)$이기 때문에

$$S(x) \equiv \sum_{i=0}^{n-1} \frac{-R_i}{x - \alpha^{-i}} \pmod{x^{n-k}} \tag{7}$$

만약 $\sum_{i=0}^{n-1} \frac{R_i}{x - \alpha^{-i}} \equiv 0 \pmod{x^{n-k}}$이면 $R(x) = \sum_{i=0}^{n-1} R_i x^i$이다.

$F_i + E_i$에 의하여 R_i를 교체하면

$$\begin{aligned} S &\equiv \sum_{i=0}^{n-1} \frac{-E_i}{x - \alpha^{-i}} \pmod{x^{n-k}} \\ &\equiv \sum_{i \in \varepsilon} \frac{-E_i}{x - \alpha^{-i}} \pmod{x^{n-k}} \end{aligned}$$

오류 ε의 위치 집합을 찾고 오류값 $\{E_i : i \in \varepsilon\}$에 대응을 위하여 두 다항식을 정의했다 첫 번째는 $i \in \varepsilon$인 α^{-i}값을 근으로 하는 다항식 "오류 위치 다항식(error locator polynomial)"이다

$$\sigma(x) = \prod_{i \in \varepsilon} (x - \alpha^{-i}) \tag{8}$$

근들을 찾음으로 해서 다항식 $\sigma(x)$을 결정할 수 있다면 오류 그 자체를 찾을 수 있다.

두 번째 다항식인 "오류 평가(error evaluator) 다항식은 (9)와 같다.

$$\omega(x) = \sum_{i \in \varepsilon} E_i \prod_{i \in \varepsilon,\ i \ne i} (x - \alpha^{-i}) \tag{9}$$

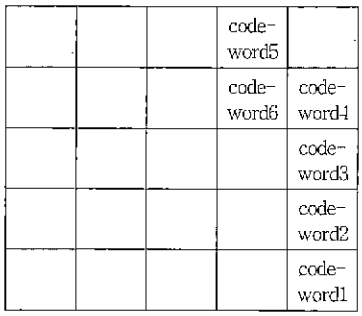
[n, k] RS code는 기껏해야 (n-k)/2 오류를 수정하기 때문에 $|\varepsilon| = \deg(\sigma) \le (n-k)/2$로 가정할 수 있다. ω는 $|\varepsilon| - 1$ 차수의 다항식의 합이기 때문에 $\deg(\omega) \le |\varepsilon| - 1$이다.

$$E_i = \frac{\omega(\alpha^{-i})}{\sigma'(\alpha^{-i})} \tag{10}$$

여기서 σ'는 σ의 유도체(derivative)이다.

3.3 2차원 바코드에 Reed-Solomon 알고리즘의 구현

3.2절에서 제시된 Reed-Solomon알고리즘을 앞에서 구현한 2차원 바코드 시스템에 구현했다. 2차원 바코드 시스템에 Reed-Solomon 알고리즘을 적용하여 기존 데이터를 새롭게 만들어서 저장하게 된다. 새롭게 만들어진 데이터의 저장 형태는 다음과 같다.



(그림 5) ECC 코드를 갖는 2차원 바코드 데이터 입력 형태

Reed-Solomon code를 작성하기 위하여 primitive polynomial의 차수는 8로 설정하였다. 그리고 CRSCode 클래스 내에서 GF에 대한 값을 구하고 원천 다항식(primitive polynomial)을 구한다. 이러한 정보를 이용하여 원래 데이터에 대한 인코딩 과정을 거친다. 인코딩된 코드 워드는 새로운 데이터와 새로운 데이터 길이를 갖는다 따라서 Reed-Solomon 알고리즘을 이용한 인코딩 과정에서 만들어진 새로운 데이터를 이용하여 2차원 바코드에 데이터 삽입과정을 거친다. 2차원 바코드에 데이터 입력시 주의해야 할 부분이 전체 바코드에 들어가는 데이터 용량이 일정한데 이 용량에 데이터를 모두 채우지 못할 경우에 채우지 못하는 구역에 대해서는 임의의 데이터로 패딩(padding)하는 작업에 필요하다. RS코드를 생성할 때도 이와같은 방법으로 해야한다.

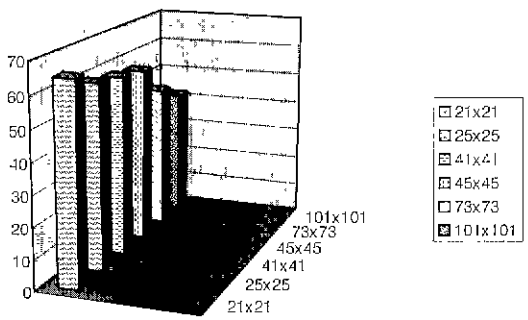
4. 실 험

본 논문에서 제시한 2차원 바코드 시스템의 오류 수정(error-correction) 방식에 대한 검증을 위하여 여러 가지 실험을 한다 실험 환경은 펜티엄 CPU를 갖는 시스템에 VC++ 5.0 프로그램 환경을 사용하였다. 그리고 디코딩을 위하여 일반 스캐너를 부착하여 스캐너로부터 이미지를 읽어들이는 방식을 취했다. 실험의 초점은 Reed-Solomon 알고리즘에 대해서 오류 복구율이 어느정도 인지에 대한 것이다. 먼저 7가지 2차원 바코드 타입에 대해서 저장할 수 있는 데이터 양은 다음 <표 1>과 같다.

<표 1> 2차원 바코드에 저장 가능한 데이터의 양

2차원 바코드 타입	저장 데이터 양 (None ECC)	저장데이타양(ECC) [Data block - ECC code]
21x21	28KB	18.2 + 9.8
25x25	40KB	24 + 16
41x41	106KB	61.8 + 44.2
45x45	127KB	72.4 + 54.6
73x73	334KB	157 + 177
101x101	672KB	282.2 + 389.8
177x177	1011KB	393.3 + 616.7

위의 <표 1>에서 보듯이 2차원 바코드 타입에 따라 저장되는 데이터 양을 구할 수 있다. 177x177 타입의 경우 3.25인치 플로피 디스켓에 저장되는 용량 정도인 1MB 가 저장 가능하다. 이 데이터 저장 용량은 ECC 코드가 포함된 경우나 포함되지 않은 경우 모두 동일하다. 그리고 ECC코드를 2차원 바코드에 저장하는 경우에 저장되는 데이터 양은 ECC코드의 길에 따라 달라진다. ECC 코드를 2차원 바코드에 저장할 경우 2차원 바코드의 타입에 따라 저장되는 데이터의 용량에 대한 전체 데이터 용량의 비를 그래프로 나타내면 다음 <그림 6>과 같다.



<그림 6> ECC 코드를 이용할 경우 2차원 바코드 타입에 따른 저장 데이터 양의 비율

<그림 6>에서 보는 바와 같이 ECC코드를 사용할 경우 2차원 바코드 시스템의 타입에 따른 데이터 양의

변화는 바코드 크기가 커질수록 저장되는 데이터 양이 줄어듬을 알 수 있다. 21x21 2차원 바코드 타입에서는 63%정도의 데이터를 저장할 수 있다. 그러나 101x101 2차원 바코드 타입에서는 43%정도 밖에 데이터 저장이 되지 않는다. 따라서 2차원 바코드 사용에서 ECC 코드의 사용으로 인하여 저장 가능한 데이터 공간에 상당한 부담을 발생시킨다는 것을 알 수 있다. 따라서 2차원 바코드에서 오류 복구 기능을 중요하게 고려해야 하는 환경에서는 ECC코드를 이용할 수 있도록 하고 오류 복구 기능이 중요하지 않은 응용 환경에서는 ECC코드를 사용하지 않도록 기능 구현이 가능하도록 한다.

그리고 오류 복구율에 대한 실험에서는 101x101 2차원 바코드 타입을 기준으로 했을 경우 2차원 바코드 손상 부분이 30~40% 정도 이상 초과되면 오류 복구가 실패함을 확인할 수 있었다.

따라서 2차원 바코드에서 ECC 코드의 사용은 오류 복구를 위한 기능 설정에서 ECC 코드를 위한 저장 공간의 낭비를 감수해야 한다. 그러나 오류 복구가 중요한 환경에서는 2차원 바코드에서 30~40%정도의 손상이 발생하더라도 데이타 복구를 해주기 때문에 필수적인 기능으로 유용하게 사용할 수 있다.

5. 결 론

본 논문은 2차원 바코드 시스템에 ECC코드를 적용하였다 2차원 바코드 시스템은 1차원 바코드 시스템과는 달리 바코드 자체가 데이터를 저장할 수 있는 능력을 갖는다. 본 논문에서는 데이터를 직접 저장 가능한 2차원 바코드 시스템을 개발했다. 2차원 바코드 시스템에 저장 가능한 데이터 양을 달리 하기 위하여 여러 가지 타입을 개발했다 2차원 바코드의 타입은 21x21, 25x25, 41x41, 45x45, 73x73, 101x101, 177x177 등 7가지가 있다. 숫자가 크면 클수록 데이터가 많이 저장된다

2차원 바코드 시스템은 1차원 바코드 시스템과는 달리 데이터를 자체 저장할 수 있는 능력을 갖고 있기 때문에 데이터 손실이 발생할 가능성이 있다. 따라서 데이터 손실에 대한 대비책을 가지고 있어야 한다. 2차원 바코드 시스템에서 데이터 손실에 대한 대비책으로 ECC 코드를 사용한다. ECC코드로 많이 사용되고 있는 것이 Reed-Solomon 알고리즘이다. 본 논문은 Reed-Solomon 알고리즘을 변형해서 사용한다.

Reed-Solomon 알고리즘을 이용하여 2차원 바코드 시스템에 데이터와 함께 코드워드 형태로 저장한다. 저장된 2차원 바코드 시스템의 데이터를 이용하여 여러 가지 실험을 하였다. 실험한 내용은 2차원 바코드 시스템에서 정상적인 데이터에 대한 저장 및 복구 기능, 정상적인 경우의 데이터 저장 용량, ECC코드를 사용하였을 경우 데이터 저장 용량의 변화, ECC코드를 사용한 데이터 오류 복구율 등이다. ECC코드를 사용하였을 경우에 정상적인 데이터 저장용량의 변화 실험에서 데이터가 많이 저장되는 2차원 바코드 시스템 타입의 경우 는 오류 복구를 위한 정보 때문에 정상적인 데이터가 40%정도밖에 저장되지 않음을 알 수 있었다. 그리고 데이터 저장 공간이 적을 경우는 60%정도의 데이터가 저장 가능함을 알 수 있었다. 그리고 오류 복구율에 대한 실험 결과 41x41 2차원 바코드 시스템을 기준으로 했을 경우 35~40%정도의 손상이 발생한 경우에 복구가 가능함을 확인할 수 있었다.

본 논문은 차세대 바코드 시스템인 2차원 바코드 시스템에 대한 새로운 개념과 설계 방향등에 대해서 언급했다. 그리고 2차원 바코드 시스템이 자체 데이터 저장 능력으로 인한 데이터 손실을 줄일 수 있는 방안으로 ECC코드의 사용 방안 및 설계 내용, 개발 내용 등에 대해서 언급했다.

참 고 문 헌

[1] 장승주, 류대현, "2차원 바코드와 우정 서비스에의 응용", pp.153-170, Postal Information Review, 1998.

[2] 오호근, 최신 바코드 기술 및 응용, 성한당, 1997.

[3] 오호근, 바코드 란 무엇인가, 한국 컴퓨터 메거진, 1994.

[4] AIM, International Symbology Specification - QR Code, Final Draft, 1997.

[5] AFMC LSO/LOA, EUCOM Demonstration Label Specification, 1998.

[6] H, Imai, *Essentials of Error Control Coding Techniques*, Academic Press, 1990.

[7] Mario Blaum, *A Course on Error-Correcting Codes*, IBM Research Center, 1997

[8] M Y, Rhee, *Error Correcting Coding Theory*, McGraw Hill, 1989.

[9] N. Glover and T. Dudley, *Practical Error Correction Design for Engineers*, Data Systems Technology, Corp., 1988

[10] P. Sweeney, *Error Control Coding, an Introduction*, Prentice Hall, 1991.

[11] Richard E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley Publishing Company, 1983.

[12] R. Goodman, R.J.McEliece, M Sayano, *Phased burst error correcting codes*, IEEE Trans. on Information Theory, Vol.IT-39, pp.684-693, 1993.

장 승 주

e-mail : sjjang@hyomin.dongeui ac.kr

1985년 부산대학교 계산통계학과 졸업(계산학)

1991년 부산대학교 계산통계학과 (계산학 이학석사)

1996년 부산대학교 컴퓨터공학과 (공학박사)

1987년~1996년 한국전자통신연구원 시스템S/W 연구실

1996년~현재 동의대학교 컴퓨터공학과 조교수

관심분야 : 운영체제, 보안, 분산처리 시스템