DBpia

논문 16-41-11-04 The Journal of Korean Institute of Communications and Information Sciences '16-11 Vol.41 No.11
http://dx.doi.org/10.7840/kics.2016.41.11.1356

LZ78 압축 데이터의 구조적 패턴에 기반한 새로운 오류 검출 알고리즘

공 명 식◆, 권 범*, 김 진 우*, 이 상 훈°

A Novel Error Detection Algorithm Based on the Structural Pattern of LZ78-Compression Data

Myongsik Gong◆, Beom Kwon*, Jinwoo Kim*, Sanghoon Lee°

요 약

본 논문에서는 LZ78 알고리즘으로 압축된 데이터의 오류 검출 알고리즘을 제안하였다. 기존의 비트 오류를 검출하는 방법들은 송신 단에서 패리티(parity) 비트를 추가하여 전송한 후, 수신 단에서 값이 '1'인 비트의 개수를 이용하여 오류를 검출하는 방법을 사용하였다. 이러한 기존의 방법들은 오류 검출을 위하여 추가적인 비트를 사용하는데 이는 데이터의 크기를 줄이는 것을 목적으로 하는 압축 데이터에 대해서는 적합하지 않은 방법이다. 따라서 본 논문에서는 LZ78 알고리즘 기반의 압축 데이터에서 추가적인 비트를 사용하지 않고 알고리즘의 구조적인 특성을 이용하여 오류를 검출하는 방법을 제안하였다. 실험을 통하여 제안하는 알고리즘의 오류 검출율에 대한 효율이 기존의 오류 검출 알고리즘에 비해 약 1.3 배 높은 효율을 가지는 것을 보였다.

Key Words : Lossless compression, Lempel-Ziv, LZ78, error detection

ABSTRACT

In this paper, we propose a novel error detection algorithm for LZ78-compressed data. The conventional error detection method adds a certain number of parity bits in transmission, and the receiver checks the number of bits representing '1' to detect the errors. These conventional methods use additional bits resulting in increased redundancy in the compressed data which results in reduced effectiveness of the final compressed data. In this paper, we propose error detection algorithm using the structural properties of LZ78 compression without using additional bits in the compressed data. The simulation results show that the error detection ratio of the proposed algorithm is about 1.3 times better for error detection than conventional algorithms.

Ⅰ. 서 론

데이터 통신에서는 이용자가 사용할 수 있는 대역폭이 제한되어 있기 때문에 효율적인 데이터 전송을 위해서는 데이터의 크기를 줄어야 한다. 따라서 데이터의 크기를 줄이는 데이터 압축에 대한 연구들이 활발히 진행되어 왔으며. 현재 데이터 압축은 통신에 관련된 거의 모든 분야에서 사용되고 있다[1-3].

데이터를 압축하는 방법에는 크게 두 가지 방법이 존재한다. 첫 번째는 데이터의 손실을 야기하면서 데

※ 이 연구는 방위사업청 및 국방과학연구소의 재원에 의해 설립된 신호정보 특화연구센터 사업의 지원을 받아 수행되었음.
◆ First Author : Yonsei University, Department of Electrical and Electronic Engineering, audlr24@yonsei.ac.kr, 학생회원
° Corresponding Author : Yonsei University, Department of Electrical and Electronic Engineering, slee@yonsei.ac.kr, 종신회원
* Yonsei University, Department of Electrical and Electronic Engineering, hsm260@yonsei.ac.kr, jw09191@yonsei.ac.kr, 학생회원
논문번호 : KICS2016-08-202, Received August 17, 2016; Revised Novemver 2, 2016; Accepted Novemver 18, 2016

www.dbpia.co.kr

이터의 전체 크기를 줄여 압축된 데이터를 가지고 원본 데이터를 완벽하게 얻어 낼 수 없는 손실 압축 방법, 두 번째는 데이터의 손실이 없이 데이터를 압축하여 압축된 데이터를 가지고 압축되기 전 원본 데이터를 완전하게 복원할 수 있는 무손실 압축 방법이 있다. 따라서 원본 데이터가 유실 되서는 안 되는 경우, 무손실 압축 방법을 통해 데이터 압축을 진행한다.

무손실 압축 방법으로 가장 널리 알려져 있고 많이 사용되는 방법으로는 Abraham Lempel과 Jakob Ziv가 개발한 Lempel-Ziv(LZ) 알고리즘이 있다[4,5]. 그 중 LZ77 알고리즘[4]의 경우 압축하고자하는 문자열이 이미 출현하였는지 확인하고 출현하였다면 상기 문자열 대신 이미 출현한 문자열에 대한 포인터의 위치와 일치하는 길이를 저장함으로서 압축하고자하는 데이터의 중복을 제거해 데이터 압축을 진행하는 알고리즘이다. 또 다른 알고리즘으로 LZ78 알고리즘이 있다[5]. LZ78 알고리즘 역시 중복된 문자열을 제거함으로서 데이터를 압축하는 방법으로 LZ77의 변형이라고 할 수 있다. LZ78 알고리즘의 경우 사전(dictionary)에 문자열들을 저장하고 문자열의 중복이 발생하면 사전에서 중복이 발생한 문자열을 검색하여 해당 문자열의 인덱스(index)를 출력한다. 이 외에 LZW, LZSS 등 LZ 알고리즘에는 LZ77, LZ78을 근간으로 하는 많은 변형된 알고리즘들이 존재하며 현재까지 많은 데이터 시스템에서 사용되고 있다[6-8].

압축된 데이터는 전송 과정에서 채널 환경의 변화, 간섭 등 다양한 요인에 의해 오류가 발생할 수 있다. 따라서 수신자는 전달 받은 데이터에서 오류가 발생 여부를 검출하는 과정을 수행한다. 압축된 데이터는 2진 부호(binary code)로 저장되어 비트(bit)열 형태로 전송에 이용된다. 비트열 전송에 있어 오류를 검출하는 대표적인 방법으로 패리티 비트(parity bit)를 이용하는 방법이 있다. 패리티 비트를 이용하는 방법은 비트열에서 1의 개수가 홀수인지 짝수인지에 따라 오류 발생여부를 판단하게 된다. 따라서 짝수개의 비트에서 오류가 발생했을 경우 패리티 비트를 사용하여 오류를 검출할 수 없게 되는 한계를 가지고 있다. 또 다른 대표적인 방법으로 해밍(Hamming)이 1950년대에 소개한 해밍 부호를 사용하는 방법이 있다. 해밍 부호는 1비트 오류에 대해서 검출 및 정정이 가능하며 2비트 오류에 대해서는 검출만이 가능하다. 하지만 3비트 이상의 오류가 발생한 데이터에 대해서는 오류 발생여부를 검출해 내지 못한다는 한계를 갖고 있다[9,10]. 또 다른 방법으로는 2차원 패리티 검사를 통하여 오류를 검출하는 블록 함 검사(block sum check)와 XOR 연산을 통한 나머지 비트를 추가적으로 전송하여 오류를 검출하는 CRC(cyclic redundancy check) 방법이 있다. 이러한 기존의 오류 검출 방법들은 모두 추가적인 비트를 함께 전송하는 방법으로서, 데이터의 크기를 줄이고자 하는 압축 데이터에 대해서는 적합하지 않은 방법이다. 따라서 본 논문에서는 기존 방법에서의 추가적인 비트를 이용하여 오류를 검출하는 방법이 아닌 LZ78 알고리즘의 특성을 이용하여 LZ78 알고리즘으로 압축된 데이터의 오류를 검출할 수 있는 방법을 제시한다.

2절에서는 기존의 오류검출 알고리즘들(패리티 체크, 해밍 부호)에 대하여 소개한다. 3절에서는 LZ78 알고리즘 및 LZ78 압축 데이터가 갖는 구조적 패턴과 이를 이용한 오류검출 알고리즘을 제안한다. 4절에서는 제안하는 오류검출 알고리즘과 기존 기술들과의 오류 검출율을 비교 분석한다. 5절에서는 본 논문의 결론을 기술한다.

Ⅱ. 기존의 오류검출 알고리즘

압축된 데이터는 압축과정에서의 오류, 전송과정에서의 누락 등 다양한 이유로 오류가 발생할 수 있다. 기존의 통신 과정에서 오류를 검출하는 방법들에는 대표적으로 패리티비트를 이용하는 방법과 해밍 부호를 이용하는 방법이 있다. 기존의 방법들은 오류가 나는 비트 수에 따라서 오류를 검출하지 못할 수도 있다는 한계를 가지고 있다. 본 절에서는 패리티 체크와 해밍 부호를 이용한 오류 검출 방법에 대해서 소개한다.

2.1 패리티 체크

패리티 체크는 패리티 비트를 이용한 오류검출 방법이다[9,10]. 패리티 비트는 어떤 비트 시퀀스의 오류를 검출하기 위한 목적으로 추가되는 비트이다. 패리티 비트의 값은 비트 시퀀스에서 '1'의 개수가 짝수가 되도록 정해진다. 그림 1의 (a) 는 임의의 비트에 대하여 패리티 비트가 추가되는 과정이다. 그림 1의 (a) 의 비트들은 모두 '1'의 개수가 짝수가 되도록 패리티 비트가 추가된다. 이런 비트들을 수신하였을 때 '1'의 개수가 홀수개로 수신되었을 경우 어떠한 원인에 의해 오류가 발생했음을 검출할 수 있다. 하지만 패리티 체크는 짝수개의 비트열에서 오류가 발생하거나 값이 '0'인 비트의 누락이 발생했을 경우는 오류를 검출하지 못한다. 이러한 한계를 그림 1의 (b)에서 확인할 수 있다.

www.dbpia.co.kr

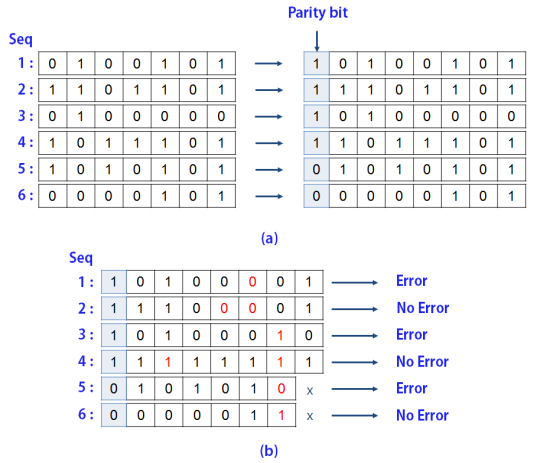

그림 1. (a) 임의의 비트 시퀀스에 패리티 비트를 추가하는 과정, (b) 패리티 체크, 빨간색은 오류 비트를, 'x' 표시는 누락된 비트를 나타낸다.
Fig. 1. (a) From Random bit sequence to bit sequence added parity bit, (b) Parity check, The red bit is error bit and 'x' is a omission.

2.2 해밍 부호

해밍 부호는 여러 개의 패리티 체크 비트를 이용한 것으로 1비트 이상의 오류는 검출하지 못하는 패리티 비트의 한계를 극복하기 위해 제안된 방법이다. 비트 시퀀스에서 데이터 비트와 패리티비트의 크기에 따라 다양한 해밍 부호가 존재한다. 대표적으로 상용되는 해밍 부호는 (3,1), (7,4), (15,11), (31,26) 해밍 부호 방식이 있다. 앞의 인덱스는 단위 비트 시퀀스의 크기를 나타내고 뒤에 나타나는 인덱스는 단위 비트 시퀀스에서 실제 정보를 나타내는 데이터의 크기를 나타낸다.

대부분 아스키(ASCII) 문자에서 한 비트열은 8비트이고 그 중 7비트가 실제 데이터이므로 해밍 부호라 하면 일반적으로 (7,4) 해밍 부호를 의미한다. 이는 4개의 비트 시퀀스에 대해 3개의 패리티 비트를 추가하기 때문에 붙여진 이름이다. 이처럼 해밍 부호는 여러 개의 패리티 비트를 이용하여 1비트 오류에 대해서는 오류를 검출하는 것뿐만 아니라 해당 오류를 수정할 수 있다. 또 2비트 오류에 대해서 오류 검출이 가능하다. 해밍 부호에서 2의 제곱에 해당하는 비트 즉, $2^k (k=0,1,2,3,...)$ 번째에 해당하는 비트들이 패리티 비트가 된다[9,10].

그림 2의 빨간색으로 된 비트들이 패리티 비트가 된다. $i(=2^k)$ 번째 패리티 비트를 p_i 라고 하면 p_i는 2^i 번째 비트부터 수식 (1)[9,10]과 같은 조건을 만족하는 비트열의 패리티 비트가 된다.

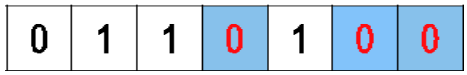

그림 2. '0111' 에 대한 해밍 부호의 패리티 비트
Fig. 2. Parity bit for '0111' in Hammin code.

$$n2^i+(m-1) \quad (m은\ 1 \le m \le 2^i\ 인\ 자연수, n은\ 자연수) \tag{1}$$

즉, p_1 은 1, 3, 5 ,7, 9, ...번째에 해당하는 비트들의 패리티 비트가 되고, p_2 은 2, 3, 6, 7, 10, 11, ...번째에 해당하는 비트들의 패리티 비트가 된다. 그림 3은 비트 시퀀스 0111에 대한 해밍 부호화 과정이다. 빨간 색 비트는 패리티 비트를, 파란색 비트는 각 패리티 비트를 결정하는 비트들을 나타낸다. 각 패리티 비트에 해당하는 비트들의 시퀀스의 1의 개수를 확인함으로서 오류 발생여부를 알 수 있다.

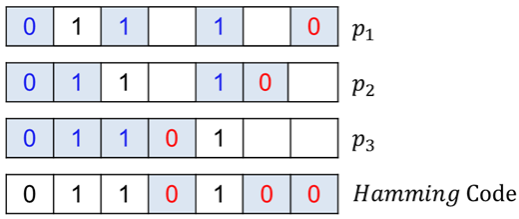

그림 3. '0111' 의 해밍 부호화
Fig. 3. Hamming coding for '0111'.

Ⅲ. LZ78 알고리즘 및 제안하는 LZ78 압축 데이터 오류검출 알고리즘

3.1 LZ78 알고리즘

LZ78 알고리즘으로 압축되는 데이터의 형태는 튜플(tuple) 형태로 이루어져 있다. 본 논문에서는 LZ78 알고리즘으로 압축된 데이터의 i번째 튜플의 첫 번째 인덱스를 A_i, 두 번째 인덱스를 B_i 라고 칭한다. 따라서 LZ78 알고리즘의 출력은 (A_i, B_i) 형태의 튜플들의 행으로 이루어져 있다. 이 때 A_i는 사전에서 매칭된 인덱스 값이 저장되고 B_i는 중복되는 문자열 바로 뒤의 하나의 문자가 저장 된다[4]. 설명의 편의를 위해, 본 논문에서는 압축하고자 하는 문자열을 Input Stream으로 명명하고, Input Stream을 LZ78알고리즘을 통하여 압축한 결과를 Output Stream이라 한다.

그림 4 는 LZ78 알고리즘으로 문자열이 압축되는 과정에 대한 예제를 보여준다. LZ78의 초기 사전에는

www.dbpia.co.kr

아무런 데이터도 저장되어 있지 않다. 따라서 첫 번째 문자는 사전에서 매칭되는 정보가 없기 때문에 Output Stream의 A_1 값에는 0이 출력되게 되며 B_1에는 첫 번째 문자인 'a' 값이 저장되게 된다. 이 후 Input Stream을 탐색해 나가는 과정에서 사전에 저장된 문자열이 출현했을 경우 사전에서 해당 문자열에 해당하는 인덱스를 Output Stream의 A_i 값에 저장하고 문자열 바로 뒤의 하나의 문자를 B_i 값에 저장하게 된다. Input Stream을 탐색하는 중에 사전에 없는 문자가 출현하면 Output Stream의 A_i 값에는 0이 저장되게 되고 B_i의 값에는 출현한 문자가 저장되게 된다. 이러한 방법으로 중복된 문자열을 없애주기 때문에 Input Stream의 길이가 길수록 이상적인 압축이 가능해 지며 길이가 짧거나 중복되는 문자열이 발생하지 않으면 압축효과가 미비할 수도 있다.

LZ78 알고리즘으로 압축된 데이터의 압축해제 방법은 압축을 하는 과정의 역순으로 진행된다. 압축을 해제할 때에는 압축할 때 만들어지는 사전정보가 필요 없다. 하지만 압축을 해제하는 과정에도 사전을 만들어 가면서 압축해제를 진행한다. 압축해제 과정에서는 Output Stream의 튜플을 보고 새로운 사전을 만들어 가면서 압축해제를 진행한다. A_i의 값이 0 이면 B_i 값을 출력한 후 사전에 저장하고, A_i의 값이 0 이 아니면 사전에서 A_i값에 해당하는 문자열과 B_i를 출력하고 출력된 총 문자열을 사전에 저장한다. 계속하여 모든 튜플을 읽어 들이고 나면 압축되기 전의 Input Stream 이 복원되게 된다.

그림 4. LZ78 압축 알고리즘
Fig. 4. LZ78 compression algorithm

3.2 제안하는 LZ78 압축 데이터 오류 검출 알고리즘

본 논문에서는 LZ78 알고리즘으로 압축된 데이터의 구조적 패턴을 이용한 오류 검출 알고리즘을 제안한다. 기존의 데이터 오류 검출 방식은 오류가 나는 비트 수에 따라 오류 검출 여부가 의존적인 방식이었으나, 본 논문에서 제안하는 알고리즘은 LZ78 알고리즘으로 압축된 데이터의 구조적 패턴을 이용하는 것이기 때문에 오류가 나는 비트 수에 상관없이 오류를 검출할 수 있다는 장점을 갖는다.

LZ78 알고리즘은 3.1절에서 설명한 것처럼 중복되는 문자열을 사전에 정하고 사전 정보를 통해 중복되는 문자열을 제거하는 방식으로 압축을 진행한다. 이런 진행 과정에서 데이터는 다음과 같은 4가지 특성을 가지고 있다. 1) LZ78 알고리즘의 Output Stream의 첫 번째 튜플의 A값 즉 A_1은 항상 0의 값을 가진다. LZ78 알고리즘에서 초기 사전에는 아무런 정보가 들어있지 않기 때문에 A_1 값은 항상 0이다. 2) i번째 튜플의 A값 즉 A_i값은 $i-1$값보다 클 수가 없다. 사전에 최대한 많은 문자열이 저장되는 경우 즉, 중복되는 문자열이 발생하지 않아 모든 문자가 사전에 저장되면서 압축이 진행된다고 하더라도 사전에는 $i-1$개의 문자열 정보가 저장되기 때문에 A_i의 값은 $i-1$을 넘을 수 없다. 3) 중복되는 튜플이 발생할 수 없다는 것이다. 한번 출현한 문자열은 사전에 저장되기 때문에 다시 등장했을 때는 반드시 사전에서 매칭이 되게 된다. 따라서 튜플 (A_i, B_i)의 값들이 모두 같은 튜플은 존재할 수 없다. 4) A_i의 값이 0인 튜플의 개수는 128개를 넘을 수 없다. 즉, 사용하는 코드(ASCⅡ)가 나타낼 수 있는 문자의 개수를 넘을 수 없다. A_i의 값이 0인 것은 사전에 문자열이 저장되어 있지 않다는 것을 뜻하고 A_i값이 0이면 길이가 1인 문자가 사전에 저장된다. 따라서 사용하는 코드로 표현할 수 있는 문자가 모두 사전에 저장되게 되면 더 이상 A_i의 값이 0인 튜플이 나올 수 없다. 1) - 4)의 특성을 수식화하면 다음과 같이 된다.

$$A_1 = 0 \tag{2}$$

$$A_i \le i-1 \,(\text{for i} = 1, 2, 3, ..., \text{N}) \tag{3}$$

$$\text{if } i \ne j, then\,(A_i, B_i) \ne (A_j, B_j), \quad (\forall\, i, j \in 1, 2, 3, \ldots, N) \tag{4}$$

$$128 > Num.\,of\ A_i = 0 \quad (\text{for i} = 1, 2, ..., \text{N}) \tag{5}$$

이러한 특성은 LZ78 알고리즘으로 압축된 데이터라면 항상 만족해야 한다. 따라서 LZ78 알고리즘으로 압축한 데이터가 위의 특성을 하나라도 만족하지 않는다면 이는 오류가 발생했음을 뜻한다. 이를 이용하

www.dbpia.co.kr

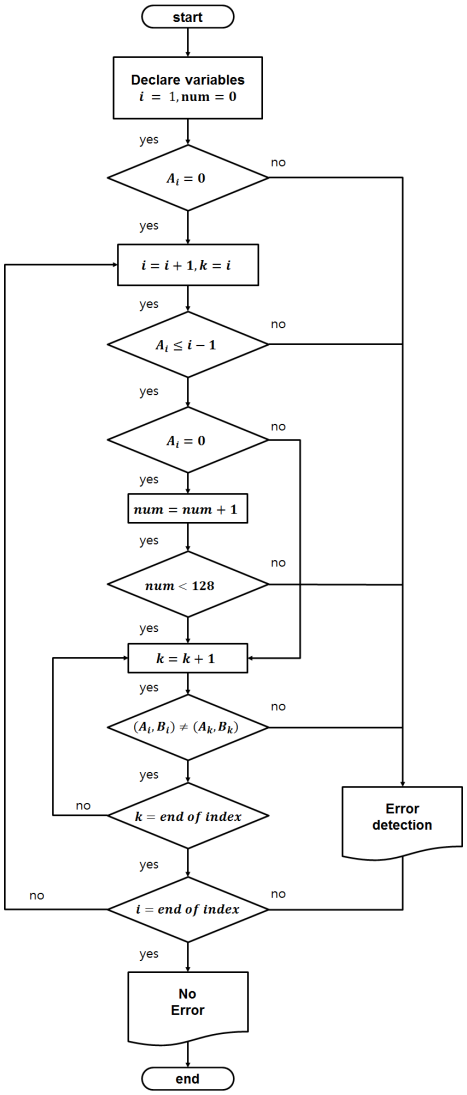

그림 5. 제안하는 오류 검출 알고리즘의 순서도
Fig. 5. Flow chart for proposed error detector

여 LZ78 알고리즘으로 압축된 데이터의 오류를 검출하는 과정은 그림 5와 같다. 그림 5의 오류 검출 과정을 통하여 LZ78 알고리즘으로 압축된 데이터의 오류 검출을 할 수 있다.

Ⅳ. 실 험

본 논문에서는 제안하는 알고리즘과 기존의 패리티 체크, 해밍 부호 오류검출 알고리즘의 오류 검출 효율을 비교하는 실험을 진행하였다. 실험에서 사용된 프로그램들은 python 언어로 작성되었으며, Window10, intel i5-4690 (3.5GHz) 환경에서 실험을 진행하였다.

각 실험은 300번의 임의의 비트 오류를 발생시켜 검출하는 경우가 몇 번인지에 대해서 측정하게 된다. 이 때 오류가 발생하는 비트 수와 오류가 발생하는 튜플의 수를 바꾸어 가며 실험을 진행하였다. 오류 검출율은 기존 방법들의 추가적인 비트의 영향력을 고려하기 위하여, 실제 input 대비 전송되는 비트열의 길이의 비율을 곱하여 normalize 한 결과로서 정의 하였다. 즉, error detection ratio = 실제 검출 비율 * (input 비트열 길이) / (전송 시 비트열 길이) 의 식을 통하여 정의 하였다.

비교하는 3가지 방법, 패리티 체크, 해밍 부호 그리고 제안하는 알고리즘 은 비트열의 길이에 따라 오류 검출에 추가되는 비트의 수가 다르기 때문에 오류검출 효율을 비교할 수 있도록 단위 비트 당 오류 검출율을 측정하여 비교하였다.

4.1. 오류 비트 개수에 따른 실험 결과

그림 6은 비트열의 길이를 11로 하여 오류 비트의 수에 따른 오류 검출율을 그래프로 나타낸 것이다. 패리티 체크의 경우 홀수의 오류 비트가 발생했을 때는 단위 비트 당 오류 검출율이 우수하지만 짝수의 오류 비트가 발생하면 오류를 검출하지 못한다. 따라서 패리티 체크의 경우는 오류검출에 사용하기에 적합하지 않다. 해밍 부호의 경우 3비트 이상의 오류에서는 약간의 효율이 떨어진다. 이는 해밍 부호의 경우 1, 2 비트 오류에 대해서는 완벽하게 오류를 검출해 내지만 3 비 비트 이상의 오류에 대해서 오류를 검출하지 못하는 경우가 발생하기 때문이다. 제안하는 알고리즘의 경우 오류 비트가 늘어날수록 오류 검출율의 효율이 증가하게 된다. 2비트 이상의 오류에서는 해밍부호의

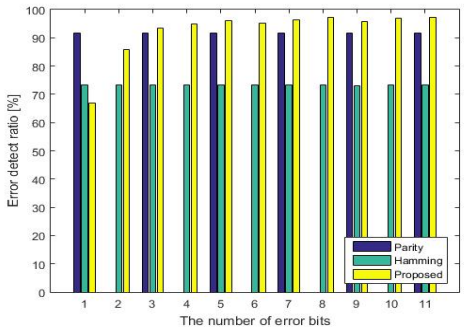

그림 6. 1튜플 오류에서의 오류 비트 수에 따른 단위 비트 당 오류 검출률, 사용한 비트열의 길이 : 11
Fig. 6. error detect ratio per bit through number error bit in 1 tuple error, length of bit sequence : 11

www.dbpia.co.kr

효율보다 더욱 높은 효율로 오류를 검출한다. 압축하고자 하는 텍스트의 길이가 길어져 사용하는 비트열의 길이가 더욱 커진다면 많은 수의 오류 비트가 발생할 가능성이 커지고 많은 수의 오류 비트가 발생할수록 제안하는 알고리즘의 효율은 더욱 더 좋아진다.

4.2 비트열의 길이에 따른 실험 결과

그림 7은 오류 비트의 개수에 따른 결과를 비트열 길이에 따라 나타낸 실험 결과이다. 그림 7의 (a) 는 비트열 길이가 7일 때의 그래프이다. 그림 7의 (a) 에서는 제안하는 알고리즘의 효율이 해밍 부호의 효율을 넘어서지 못한다. 그림 8의 (b), (c) 에서부터 제안하는 알고리즘의 효율이 해밍 부호의 효율보다 높아지는 경향을 보이는 것을 알 수 있다.하지만, 오류가 1비트에 해당할 때는 효율이 해밍코드 보다는 떨어지

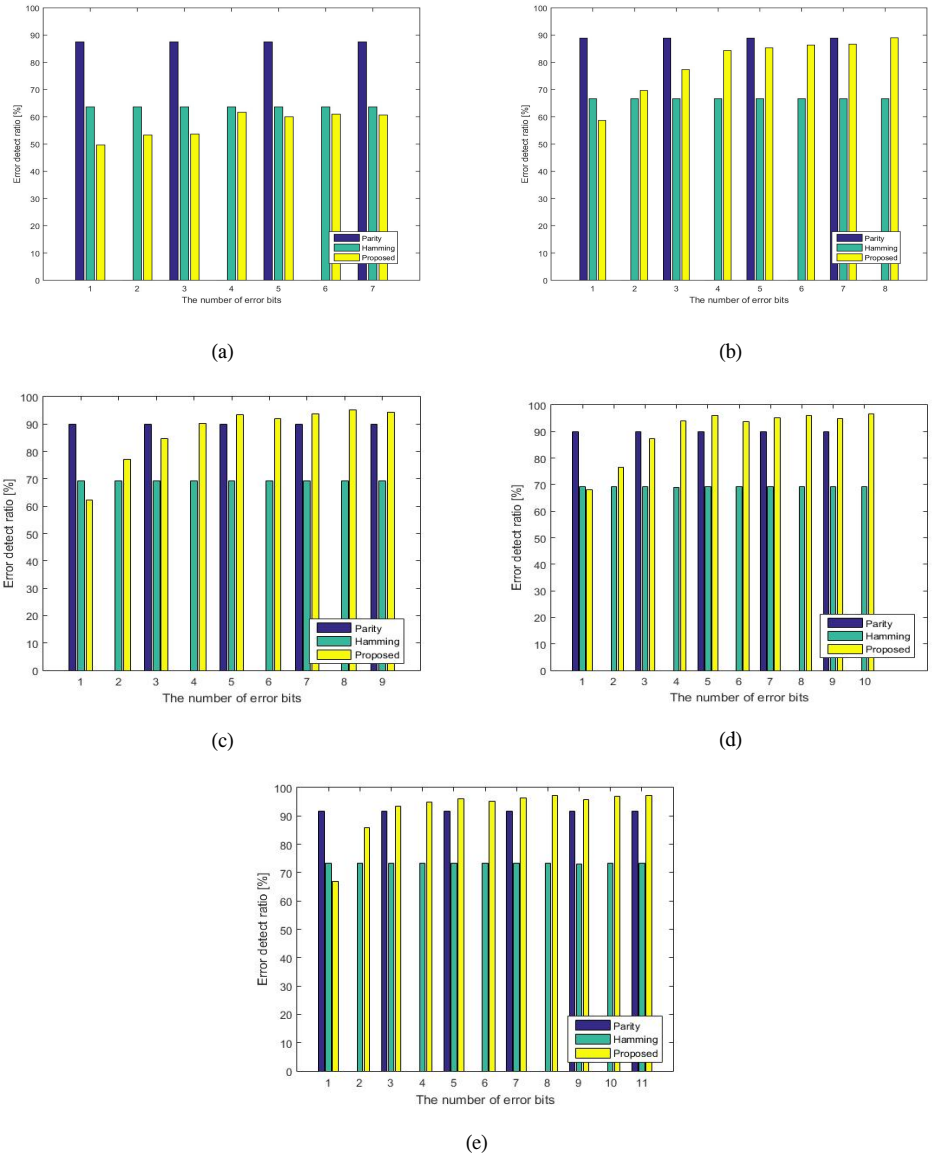

그림 7. 비트열의 길이에 따른 결과 그래프, (a) 비트열의 길이 : 7, (b) 비트열의 길이 : 8, (c) 비트열의 길이 : 9, (d) 비트열의 길이 : 10, (e) 비트열의 길이 : 11

Fig. 7. error detect ratio per bit through number error bit in different length of bit sequence in 1 tuple error, (a) length of bit sequence : 7, (b) length of bit sequence : 8, (c) length of bit sequence : 9, (d) length of bit sequence : 10 (e) length of bit sequence : 11

www.dbpia.co.kr

는 것으로 나타난다. 그림 7의 (d)에서 볼 수 있듯이 비트열의 길이가 10일 때부터

제안하는 알고리즘의 효율이 해밍 부호의 효율이 대부분의 경우에서 우수함을 알 수 있다. 즉 제안하는 알고리즘은 사용하는 비트열이 길수록, 오류 비트가 많을수록 더욱 더 좋은 효율을 나타낸다. 실험을 통해서 보이듯 사용하는 비트열의 길이가 길수록 제안하는 알고리즘의 효율이 높아지는 것을 알 수 있다. 이는 사용하는 압축 데이터의 튜플의 수가 많을수록 즉, 압축하고자 하는 데이터의 크기가 클수록 제안하는 알고리즘의 오류검출 효율이 해밍 부호의 효율보다 좋다는 것을 의미한다.

그림 6, 7의 결과로부터, 제안하는 알고리즘은 오류가 발생하는 비트가 많을수록, 사용 비트열의 길이가 길수록 높은 효율을 가지게 되는 것을 알 수 있다. 오류 비트 수가 2비트 이하 일 때 해밍코드의 효율이 더 좋지만 실제 전송 환경에서는 일반적인 평문의 경우 비트열의 길이가 길고 오류 비트의 수도 더 많은 경우가 대부분이므로 제안하는 방법의 활용성이 기존의 방법 보다 더 뛰어나다고 할 수 있다. 또한 기존의 방법들은 모두 추가적인 비트를 추가하여 오류를 검출시키는데, 이는 데이터의 크기를 줄이고자 하는 압축 데이터에는 적합한 방법이 아니다. 추가적인 비트의 수까지 고려하여 비트길이당 오류 검출 효율을 생각하면 제안하는 방법이 더욱더 효율적임을 알 수 있다. 비트 시퀀스의 길이가 7비트 이하의 비트열에서는 패리티 비트와 같은 기존의 방법을 사용하고, 7비트 이상의 비트열에서는 제안하는 방법을 이용한다면 최적의 효율로 오류를 검출 할 수 있을 것으로 생각된다.

V. 결 론

본 논문에서는 LZ78 알고리즘으로 압축된 데이터의 구조적 패턴을 이용한 오류검출 알고리즘을 제안하였다. 기존의 비트열 오류검출 알고리즘인 패리티 체크 와 해밍 부호는 오류를 검출하기 위해 추가적인 비트가 필요하며 오류가 발생하는 비트 수에 영향을 받는다는 한계를 가지고 있다. 기존의 방법은 추가적인 비트를 사용하므로 데이터의 크기를 줄이는 것에 목적을 갖는 압축 데이터에 적합하지 않는다. 그에 반해 제안하는 알고리즘은 추가적인 비트 없이 오류를 판별하고 오류가 발생하는 비트수에 독립적인 방법으로서 압축 데이터에 적합하다. 실험을 통해 제안하는 방법이 비트열의 길이가 길수록, 오류 비트의 수가 많을수록 더 효율적임을 알 수 있었다. 다음 연구에는 다른 무손실 압축 알고리즘(LZSS, LZ4, LZW) 에 대해서도 추가적인 비트 없이 오류를 효율적으로 검출할 수 있는 방법의 연구를 진행하여, 압축 데이터를 자동으로 판별 한 후 오류를 검출하는 연구를 진행할 계획이다.

References

[1] J. Park, T.-W. Ban, and B. C. Jung, "A compressed sensing-based signal recovery technique for multi-user spatial modulation systems," *J. KICS*, vol. 39, no. 7, pp. 424-430, Jul. 2014.

[2] I. Lee, H. Kim, T. Oh, and S. Lee "Study on low delay and adaptive video transmission for a surveillance system in visual sensor networks," *J. KICS*, vol. 39, no. 5, pp. 435-446, May 2014.

[3] Thu L. N. Nguyen, H. Jung, and Y. Shin, "A signal detection and estimation method based on compressive sensing," *J. KICS*, vol. 40, no. 6, pp. 1024-1031, Jun. 2015.

[4] J. Ziv and A. Lempelm "A universal algorithm for sequential data compression," *IEEE Trans. Inf. Theory*, vol. 23, no. 3, pp. 337-343, 1977.

[5] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inf. Theory*, vol. 24, no. 5, pp. 530-536, 1978.

[6] S. Jung, D.-I. Seo, and B.-R. Moon, "Performance improvement of LZ77 algorithm using a strategy table and a genetic algorithm," *J. KIISE*, vol. 31, no. 12, pp. 1628-1638, Dec. 2004.

[7] T. A. Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6. pp. 8-19, 1984.

[8] J. A. Storer and T. G. Szymanski, "Data compression via textual substitution," *JACM*, vol. 29, no. 4, pp. 928-951, Oct. 1982.

[9] D. MacKay, *Information theory, inference and learning algorithms*, Cambridge University Press, 2003.

[10] T. K. Moon, *Error correction coding*,

www.dbpia.co.kr

Mathematical Methods and Algorithms, John Wiley & Sons, 2005.

공 명 식 (Myongsik Gong)

2016년 2월 : 연세대학교 전기전자공학과 졸업
2016년 3월~현재 : 연세대학교 전기전자공학과 석박사통합과정
<관심 분야> 이미지 프로세싱

권 범 (Beom Kwon)

2012년 2월 : 숭실대학교 정보통신전자공학과 졸업
2012년 3월~현재 : 연세대학교 전기전자공학과 석박사통합과정
<관심 분야> 통신네트워크

김 진 우 (Jinwoo Kim)

2016년 2월 : 홍익대학교 전자전기공학과 졸업
2016년 3월~현재 : 연세대학교 전기전자공학과 석박사통합과정
<관심 분야> 이미지 프로세싱

이 상 훈 (Sanghoon Lee)

1989년 2월 : 연세대학교 전기전자공학과 학사
1991년 2월 : KAIST 전기전자공학과 석사
2000년 1월 : 텍사스 대학교 전기전자공학과 박사
2003년 2월~2007년 3월 : 연세대학교 전기전자공학과 조교수
2007년 4월~2012년 2월 : 연세대학교 전기전자공학과 부교수
2012년 3월~현재 : 연세대학교 전기전자공학과 정교수
<관심 분야> 이미지 프로세싱, 컴퓨터 비전, 화질평가, 통신네트워크

www.dbpia.co.kr