JBIG2 부호화에서의 한글의 효율적 처리에 관한 연구

정회원 강 병 택*, 김 현 민**, 고 형 화**

A Study on Effective Processing of Hangul for JBIG2 Coding

Byung-taek Kang*, Hyun-min Kim**, Hyung Hwa Ko** *Regular Members*

요 약

본 논문은 차세대 2진 영상 압축 표준인 JBIG2를 한글문서에 적용시켰을 때 압축률을 개선할 수 있는 방법을 제안하였다. 한글은 영어와 같은 로마 계열 문자와 달리 한 문자가 여러 개의 자소로 구성된다. 따라서 전송해야 할 위치정보가 많아지는 단점이 있다. 이런 단점을 개선하기 위해서는 여러 개의 자소단위 심벌을 의미 있는 문자 단위 심벌로 결합하는 방법을 제안하였다. 제안한 한글 문자 결합 알고리즘은 산술부호화보다는 허프만 부호화에서 더욱 많은 개선을 얻을 수 있었다. 무손실 압축의 경우 제안한 방법을 적용하지 않은 경우와 비교할 때, 허프만부호화의 경우 4.5~16.7(%), 산술부호화의 경우 2.9~10.4(%) 정도의 압축률 개선을 얻을 수 있었다. 유손실 압축의 경우는 허프만부호화의 경우 3.7~17.0(%), 산술부호화의 경우 2.1~10.5(%) 정도의 압축률 개선을 얻었다.

ABSTRACT

In this paper, we propose a method to improve JBIG2 compression ratio which can be applied to Hangul text. Hangul character is composed of a few symbols which is called JASO, which needs inevitable increase of position information to be transmitted. To reduce this disadvantage, we have proposed an algorithm that generate aggregated symbol in combination of JASO symbols. Proposed algorithm shows better performance in Huffman coding than in arithmetic coding. In lossless coding, proposed algorithm showed 4.5~16.7(%) improvement for Huffman coding and 2.9~10.4(%) improvement for arithmetic coding. In lossy coding, proposed algorithm showed 3.7~17.0(%) improvement for Huffman coding and 2.1~10.5(%) improvement for arithmetic coding.

Ⅰ. 서 론

통신 수단이 발전하면서 문서의 압축 저장과 전송 요구가 늘어가고 있다. 그런데, 이러한 문서의 저장과 전송은 압축 방법을 요구하게 된다. 근래에 2진 영상을 대상으로 하는 표준 압축 알고리즘 (JBIG2) [1]은 동영상을 대상으로 하는 MPEG-4, MPEG- 7과 정지영상을 대상으로 하는 JPEG2000 등과 마찬가지로 표준화가 진행 중에 있다.

팩스 및 문서의 저장에 주로 응용되었던 기존의 2진 영상의 표준 압축 알고리즘은 MH(Modified Huffman), MR(Modified READ)[2], MMR (Modified Modified READ)[3] 그리고 JBIG [4] 순으로 발전되었다. MR과 MMR은 문서의 이전 라인과 현재라인의 상대적 위치를 이용하여 부호화를 행하며 각각 G3, G4 팩스에 의해서 사용되었다. JBIG은 MR과 MMR이 허프만 부호화를 사용하는 것과 달리 산술부호화를 사용하였다. 또한 스캔라인 순서가 아닌 부호화 화소의 주변화소들로부터 템플릿을 이용하여 콘텍스트를 구해 산술부호화의 확률 평가를 이용하여 압축을 행하였다.

* 삼성전기주식회사 시스템 사업부 연구실 (kangbt@samsung.co.kr)
** 광운대학교 전자통신공학과 영상처리 연구실 (minius@explore.kwangwoon.ac.kr, hhkoh@daisy.kwangwoon.ac.kr)
논문번호 : 00024-0117, 접수일자 : 2000년 1월 17일
※ 이 연구는 1998학년도 연구년 및 1999학년도 교내 학술 연구비 지원에 의하여 이루어 졌음.

JBIG2는 AT&T에 의해 제안된 SPM(Soft Pattern Matching)과 제록스(Xerox)에서 제안한 심벌에 기반하는 기법[5] 및 IBM에서 제안한 BST 알고리즘[6] 등에 기반하여 JBIG(Joint Bi-level Image Experts Group)위원회에 의해 표준화가 진행중이고 현재 거의 마지막 단계인 FDIS(Final Draft International Standards)상태에 있다[1].

JBIG2 압축 방법의 개요는 다음과 같다. 전형적인 text 페이지에는 많은 반복적인 문자들이 존재한다. 각 문자의 모든 화소들을 그대로 부호화하기보다는 반복되는 문자들의 대표치를 비트맵으로 사전(dictionary)에 저장한다. 이러한 비트맵을 심벌이라고 부른다. JBIG2는 추출된 심벌의 집합인 사전에 있는 정보들을 이용하여 문서를 압축한다. 만약 현재 심벌이 사전에 존재하지 않는 심벌인 경우에는 이를 사전에 추가하고 인덱스와 위치정보를 보내고, 심벌이 사전에 이미 등록이 되어 있는 경우에는 사전에 추가되지 않고 심벌의 인덱스와 위치정보만을 보내는 방식으로 부호화가 이루어지게 된다. 이렇게 함으로써, 반복적으로 나타나는 심벌에 대하여 전송 비트량을 줄일 수 있다.

JBIG2의 중요한 특징을 정리해 보면, 첫째, JBIG2는 SPM 알고리즘을 사용한다. 이 방법은 Johnsen[7]등에 의해 제안된 PMS (Pattern Matching & Substitution) 알고리즘의 오매칭 에러를 해결하기 위해서 Witten[8]과 Howard[9]에 의해 제안된 알고리즘으로 오매칭으로 생기는 에러에 대한 정련(refinement) 정보를 추가해줌으로써 해결했다.

둘째, JBIG2는 압축률과 부호화 및 복호화 시간과 사용 메모리 등을 감안한 복잡도 등을 고려하여 응용영역에 맞는 Profile을 제공해 주는 장점을 가지고 있다. 즉, 문서 저장(archiving)을 위한 고압축률/고복잡도 profile, WWW을 위한 중압축률/중복잡도 profile, Fax 전송을 위한 저압축률/저복잡도 profile등이 있다. 각 Profile별로 사용하는 부호화 요소를 선택 가능하게 하였다.

셋째, JBIG2는 기존 2진 영상 압축 표준에 비해 다양한 부호화를 허용한다. MH, MR, MMR이 허프만 부호화기만을 사용하고, JBIG1이 산술부호화기만을 사용하는 것에 반해 JBIG2는 허프만 부호화와 산술부호화기를 선택적으로 사용한다. 이것은 사용자로 하여금 압축률은 낮지만 빠른 실행을 요구하는 응용의 경우에는 허프만 부호화를, 부호화 시간보다는 높은 압축률을 요구하는 경우에는 산술부호화기를 사용할 수 있도록 하고 있다.

넷째, 여러 페이지로 구성된 문서에 대해서도 이전 페이지에서 얻었던 심벌 사전을 다시 사용할 수 있으므로 한 페이지만을 대상으로 압축을 행하는 기존 방식에 비해 페이지수가 많아질수록 압축 면에서 더 좋은 성능을 얻을 수 있다는 점이다.

본 논문에서는 이러한 JBIG2 압축방식을 한글이 포함된 2진 문서에 적용했을 때[10], 한글이 갖는 고유한 특징을 이용하여 전처리를 통해 심벌처리 과정에서 이득을 얻을 수 있음에 착안하여 압축률을 높일 수 있는 방식을 제안하였다. 본 논문의 구성은 2장에서 JBIG2의 개요를 알아보고, 3장에서 제안한 한글자소의 결합 알고리즘을 기술하였다. 4장의 실험 및 고찰에 이어 5장에서 결론을 도출하였다.

Ⅱ. 차세대 2진 영상 표준 압축 알고리즘 (JBIG2)의 개요

JBIG2의 복호기 블럭도를 그림1에 보였다. 본 논문에서는 그림1의 JBIG2 복호기 블록도 중 점선으로 둘러싸인 부분에 대해서만 다루었다. JBIG2는 다른 표준과 마찬가지로 복호기만을 표준으로 정의한다. 따라서 어떻게 부호화기를 구성하는가는 표준화되어 있지 않다. 텍스트 영역이라면 halftone과 같은 그림이 없고 순수하게 읽을 수 있는 글로만 된 영역을 말한다. 한 문서에서 이 텍스트 영역은 여러 개로 나누어 질 수도 있다. 이것은 부호화기에 의존적이다. Generic 영역은 심벌을 제외한 나머지 그래픽이나 기호 등을 부호화하는 부분이며 허프만이나 산술 부호화기에 의해 부호화하게 된다.

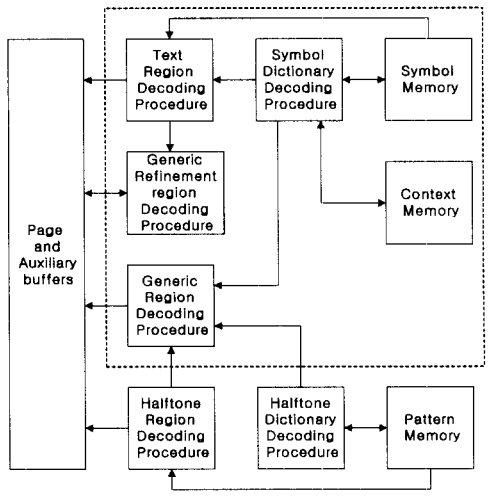


그림 1. 주요 복호기 성분들의 블럭도

1. 심벌 사전(Symbol Dictionary)부호화

JBIG2에서 심벌 사전은 추출된 심벌 비트맵들의 저장장소이다. 심벌은 사전에 등록되는 최소 단위라 말할 수 있다. 영문의 경우 "a", "b" 그리고 "ab"일 수도 있다. 한글의 경우에는 "ㄱ", "ㅏ"등이나 "가"도 될 수 있다. 문서에 잡음 성분이 많은 경우 심지어 "."과 같은 성분도 심벌이 될 수 있다. 개요에서 설명한 것처럼 JBIG2는 심벌 사전을 구성하는데 최초에는 사전이 비어 있지만, 압축될 문서에서 심벌을 추출하여 새로운 심벌로 판정될 경우 계속적으로 사전에 저장한다.

심벌 사전 부호화는 심벌 사전에 등록된 심벌 비트맵들을 어떻게 효율적으로 압축하느냐에 관한 문제이다. 이를 위해서 JBIG2는 심벌 사전의 부호화에 대해 다음과 같은 방법을 적용한다.

첫째, 심벌 사전의 구조를 정의하여 심벌의 높이별로 구분하여 클래스를 구성한다. 높이가 가장 낮은 심벌비트맵들의 클래스부터 순차적으로 부호화를 행한다. 이렇게 하면 다음 클래스는 이전 클래스의 높이의 차이만 저장하므로 좀 더 효율적인 부호화를 할 수 있게 된다

둘째, 각 클래스 내에서는 심벌 비트맵들의 폭이 가장 작은 값부터 순차적으로 부호화를 행하게 된다. 이것은 위와 마찬가지로 심벌비트맵의 폭을 차이 값만 부호화하므로 이득을 얻게 된다.

셋째, 표2 에서 보는 것처럼 각 클래스의 부호화는 두 가지 모드를 준비하고 있다. 이것은 허프만부호화와 산술부호화이다. 표2 에서 쓰여진 SDHUFF은 심벌 사전의 부호화를 위해서 허프만 부호화를 사용했는지를 나타내고, SDREFAGG은 심벌 비트맵을 산술부호화를 이용하여 부호화를 행할 경우 Aggregte 심벌이 포함되었는지를 나타낸다. 예를 들어, "the"라는 심벌을 심벌 사전에 추가 할 때 이미 심벌 사전에 "t", "h" 그리고 "e"가 있는 경우, 비트맵을 직접 저장하지 않고, 이 심벌들의 인덱스와 위치정보를 이용하여 "the"라는 aggregte 심벌을 사전에 저장할 수 있는 방법으로 비트맵을 직접 부호화할 때 보다 효율적인 부호화를 행할 수 있는데 이러한 것을 AGGREGATE 방법이라고 한다.

표2(a)는 허프만 부호화의 경우이고 이때 height class의 구조는 산술부호화를 사용한 경우의 표2(b)와 비교해서 가장 큰 특징은 collective bitmap이라고 할 수 있다. 이처럼 높이가 같은 심벌들 전체를 모아서 부호화는 행하는 이유는 MMR을 사용하기 때문에 기준라인과 현재 라인을 효율적으로 이용하기 위해서이다. 표2(b)의 경우는 산술 부호화를 사용한 경우이며 가장 큰 특징이라면 각 비트맵들을 개별적으로 부호화한다는 것이다.

2. 텍스트 영역(Text Region) 부호화

텍스트 영역 부호화는 텍스트에서 추출된 심벌이 저장된 심벌 사전내의 인덱스와 텍스트상의 수평위치(S) 및 수직위치(T)를 부호화 과정이라 말할 수 있다. 즉, 심벌의 비트맵은 앞 절에서 설명한 방법대로 심벌 사전에 저장되고, 각 심벌의 인덱스와 위치정보는 텍스트 영역 부호화를 통해 저장된다.

표 1. 심벌사전의 구조

First height class
Second height class
.....
Last height class
List of exported symbols

표 2. 높이 클래스 부호화

(a) SDHUFF=1 & SDREFAGG=0

Height class delta height
Delta width for first symbol
Delta width for second symbol
.....
OOB(out-of-band)
Height class collective bitmap

(b) SDHUFF=0 OR SDREFAGG=1

Height class delta height
Delta width for first symbol
Bitmap for first symbol
Delta width for second symbol
Bitmap for second symbol
.....
OOB

표 3은 텍스트 영역의 부호화에 사용된 구조이다. 여기서, 전체 문서를 한꺼번에 처리하려면 많은 메모리가 요구되므로 JBIG2 표준에서는 일정 크기의 STRIP 단위로 분할하여 처리하게 된다. 1개의 STRIP은 텍스트 영역을 래스터 스캔 방법으로 스캐닝한 후 SBSTRIPS 크기의 스캔 라인씩 그룹화

하여 문서를 분할하게 된다. 여기서 SBSTRIPS는 1, 2, 4, 8 값 중에서 사용자가 선택할 수 있다. 각 STRIP에는 여러 개의 심벌이 존재할 수 있다. 각 Strip들은 아무런 심벌을 가지지 않은 경우도 있으므로 표3의 Delta T를 이용하여 이전 Strip과 현재 Strip의 시작 위치를 조절하게 된다. 표4는 각 심벌의 부호를 위한 구조인데, 표4(a)는 심벌의 정련을 텍스트 영역에서 사용하지 않은 경우이다. 이 경우 심벌들은 위와 같이 수평(S)위치, 수직위치(T), 그리고 인덱스 정보만을 전송하면 된다. 표4(b)는 심벌의 정련을 텍스트 영역에서 행하는 경우이다.

텍스트 영역 부호화의 예를 그림 2에서 보였다. 그림 2(a)에서 사용된 점선들은 Strip의 시작점이다. 각 Strip안의 심벌들은 먼저 수평위치 순서대로 부호화를 하게 된다. 첫 심벌 부호화는 텍스트 영역의 비트맵의 수평위치를 기준으로 차를 부호화하게 된다. 그림 2(a)의 DFS가 이 차 값을 나타내는 값이다. 다음으로 나머지 심벌들은 이전 심벌의 위치를 기준으로 차들을 계속해서 부호화하게 된다. 그림 2(a)의 IDS가 이 값을 나타내며 SBDSOFFSET은 심벌들 간의 평균거리를 예상하여 주어진 값이다. 그리고 심벌의 수직위치에 대해서는 그림 2(b)의 "k"의 심벌의 수직위치(T)는 Strip의 시작 위치(점선)를 기준으로 부호화를 행하게 된다.

표 3. 텍스트 영역의 부호화 구조

Initial StripT value	Delta T
First Strip	First symbol
Second Strip	Second symbol
.....	.....
Last Strip	Last symbol
	OOB

표 4. 심벌의 구조

(a) 정련을 행하지 않을 경우

Symbol's 수평(S) coordinate
Symbol's 수직(T) coordinate
Symbol's ID

(b) 정련을 행할 경우

Symbol's 수평(S) coordinate
Symbol's 수직(T) coordinate
Symbol's ID
Symbol's 정련(refinement) 정보

만약 SBSTRIPS가 4라면 심벌의 수직위치는 0,1,2,3 중에 선택하게 되고, 2진수로 표시하면 00, 01, 10, 11이다. 이 값은 현재의 Strip의 시작점에서부터 아래쪽으로의 Offset값을 나타내게 된다.

Ⅲ. 제안한 한글의 심벌 결합 알고리즘

한글로 구성된 문서에서 심벌을 추출했을 경우 문자 단위 심벌이 아닌 자소 단위 심벌이 추출되게 된다. 이것은 이전의 텍스트 영역 부호화에서 설명한 방식대로 부호화를 행한다면 영문에 비해 많은 위치정보가 필요할 수밖에 없다.

예를 들어, 그림 2에서 영문의 "k"와 "W"는 둘 다 3번째 Strip에 속해 있다. 따라서 "W"의 수평위치정보는 "k"의 마지막 수평위치와 "W"의 시작 수평위치의 차로 부호화가 가능하다. 그러나 한글의 경우 "해"자를 부호화하려면 Strip 1번에 속하는 1-2 심벌은 Strip 1번에 속하는 "하"자의 1-1 심벌과 비교해야 한다. 이것은 중간에 있는 "가"의 어떤 자음과 모음도 Strip 1에 포함되지 않으므로 이 부분을 건너뛰어 1-1 심벌의 수평위치의 끝과 1-2 심벌의 수평위치의 시작점과의 차로 부호화를 해야함으로 더 많은 부호화 비트가 필요하게 된다. 현재의 그림 2(b)에선 "하"와 "해"사이에 "가"자 하나 뿐인 경우이지만 최악의 경우에는 각 라인의 시작에서부터 끝나는 부분까지의 거리를 가질 수도 있을 것이다. 또한, 수직위치에 대해서도 "하"자의 경우 1-1 심벌은 Strip1에, 2 심벌은 Strip2에, 마지막으로 3-1 심벌과 3-2 심벌은 Strip3에 각각 소속되어 있으므로 효율성이 떨어지게 된다. 이것은 영문자의 경우 "k", "W"등의 수직위치를 부호화할 때 단지 한번의 부호화로 끝나는 것과는 대조적이다. 이처럼 자음과 모음으로 구성된 한글은 로마 계열문자인 영문에 비해 위치 정보를 전송하는데 JBIG2에서는 더욱 많

그림 2. 예제 비트맵

은 비트를 할당해야만 한다. 이런 문제를 해결하기 위해서 본 논문에서는 한글의 자음과 모음 등을 결합하여 자소단위가 아닌 문자단위로 텍스트 영역을 부호화하는 알고리즘을 제안한다. "하"자의 경우 1-1, 2, 3-1 그리고 3-2 심벌들로 구성된 자음과 모음을 결합해서 의미 있는 문자 단위인 "하"자를 만들었다. 이 경우 우리 한글도 로마계열 문자인 영문과 마찬가지로 심벌의 위치에 대해 동일하게 처리할 수 있었다.

1. 상하로 대상을 결합하는 단계

자음과 모음으로 구성된 한글을 문자 단위 심벌로 결합하기 위해서 먼저 상하로 자음과 모음을 결합하게 된다. 이것은 먼저 추출한 심벌들에 대해 라인영역을 찾는 것이 필요하다. 라인 영역은 문서에서 글자들이 존재하는 부분으로 문서의 시작 위치인 왼쪽과 끝 위치인 오른쪽으로부터 검색을 시작하여 black 화소를 만나게 되면 그 라인은 글자가 존재하는 영역이 된다. 이와 같은 방법으로 검색을 하면 글자가 존재하는 부분과 존재하지 않는 부분이 차례대로 찾아지게 되고 이때 글자가 존재하는 영역을 라인영역으로 했으며 순서대로 라인영역을 저장했다. 이와 같이 텍스트 영역의 라인 영역이 얻어지면 각 라인 영역에 대해서 수평위치별로 심벌을 정렬해야 한다. 이렇게 정렬함으로써 심벌의 상하 방향으로 문자의 결합이 가능하게 된다.

그림 3은 상하로 대상 심벌을 결합하는 방법에 대한 흐름도이다. 예로 그림 2(b)에서 라인 영역은 L1과 L2사이가 된다. 이 영역에서 심벌을 추출하고 수평위치가 빠른 순서대로 심벌을 정렬하게 된다. 그림 2(b)의 비트맵을 수평위치에 따라 정렬하면 "하"의 2, 3-1, 1-1, 3-2 심벌 그리고 "가"의 3-3, 3-4 심벌 순으로 정렬된다. 정렬 순서에서 처음 만난 "하"의 2심벌에 대해 다음 3-1 심벌의 비트맵이 L과 R사이에 조금이라도 속하는 부분이 있으면 3-1 심벌도 이 심벌에 결합하게 되고 다시 L과 R값을 구하게 된다. 현재의 L과 R은 3-1 심벌의 L과 R을 포함하므로 그대로 유지하게 된다. 다시 다음 1-1 심벌에 대해서도 위와 같은 비교를 행하게 된다.

이제 2, 3-1과 1-1 심벌은 결합되어 하나의 비트맵이 되고 3-2 심벌은 L과 R사이에 포함이 되지 않기 때문에 결합을 시도하지 않게 된다

2. 중성 모음 대상 심벌을 결합하는 단계

한글의 중성 모음 중 첫 번째 단계에 의해 결합

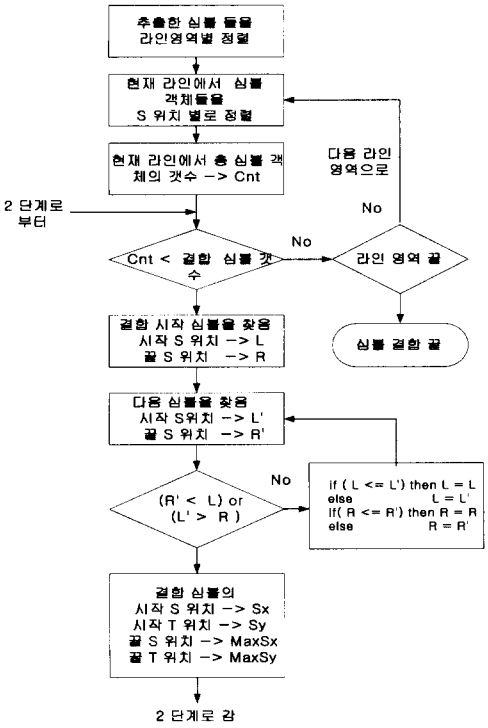


그림 3. 상하 대상 심벌을 결합하는 알고리즘

되지 않은 대상 심벌은 "ㅏ, ㅑ, ㅓ, ㅕ, ㅣ,ㅐ, ㅒ, ㅔ, ㅖ" 같은 9개의 모음이 존재한다. 이것들의 공통점은 다음과 같다.

1) 중성모음의 높이에 비해 초성의 높이가 낮다.
2) 초성의 폭과 비슷한 정도의 떨어진 위치에 중성 모음이 놓인다.
3) 중성의 높이는 폭에 비해 2배 정도 더 높다.

본 논문은 이런 근사적인 공통점을 근간으로 중성모음 대상 심벌을 찾고 첫 번째 단계에서 얻어진 심벌에 결합했다. 이 부분은 상당히 많은 연구가 필요한 부분이다. 잘못된 중성모음의 결합은 원하지 않은 심벌의 결합을 가져 오게된다. 이것은 효율적인 압축에 불리한 결과를 가져 올 수밖에 없다. 본 논문에서는 중성 모음의 조건으로는 위의 공통점 외에도 원하지 않는 중성 모음의 결합을 막기 위해 몇 가지 조건을 더했다. 그림 4에 중성 대상 심벌을 결합하는 흐름도를 보였다.

1) 초성의 높이는 중성 모음의 높이의 1/4이상이어야 한다.
2) 중성 대상 심벌의 black 화소의 연결 높이는 심벌의 높이의 7/9 정도는 되어야 한다.

첫 번째 조건은 ". 1"과 같은 너무 작은 초성 심벌에 원하지 않는 심벌이 결합하는 것을 막기 위한 것이고 두 번째 조건은 "9)"와 같이 ")"성분이 결합되는 것을 막기 위한 것이다. 즉 아래 부분으로부터 대상 심벌의 흑화소 연결 부분의 크기 값을 찾으면 "ㅏ"의 경우는 "ㅣ"부분에서 최대로 black 화소 연결 부분을 얻을 수 있다. 그러나 ")"는 black 화소 연결 부분을 아래 부분부터 위로 조사하다 보면 부분적으로 끊어져서 심벌 높이의 7/9정도까지 black 화소가 연결된 부분은 찾을 수 없을 것이다.

Ⅳ. 실험 및 고찰

컴퓨터 모의 실험은 제안한 방법을 적용했을 경우와 적용하지 않았을 경우에 대해서 성능을 비교 평가하였다. 테스트를 위해 사용된 문서는 스캐너로 취득한 한글문서 2장을 사용하였다. 한글문서를 스캔해서 실험한 이유는 표준에는 한글문서가 없기 때문에 한글문서에 대해 제안한 알고리즘의 성능을 평가하기 위해서이다. 또한, 본 알고리즘이 로마자 계열 문서에서도 무난히 동작됨을 보이기 위해 ITU-T 표준 문서인 F01, F04문서에 적용해 보았다. 한글 문서 1은 고딕체 문서이고, 한글문서 2는 명조체와 고딕체가 섞인 문서이다. 각 문서에서 200, 300, 400, 600(dpi)의 해상도에서 실험하였다. 실험은 펜티엄-Ⅲ 450 MHz 개인용 컴퓨터를 사용하였다. 한글문서 1과 2에 제안한 한글 전처리방법을 적용하지 않은 경우, 적용한 경우, 그리고 JBIG2 표준안에 규정된 REFAGG 모드로써 심벌 사전을 구성할 때, 몇 개의 심벌을 묶어서 하나의 심벌로 저장하는 AGGREGATE 방법을 적용한 경우에 대하여 실험하였다. 여기서 편의상 M1은 제안한 방식을 적용하지 않은 경우이고, M2는 적용한 경우를 의미하며 REFAGG는 심벌의 AGGRE- GATE 방식을 적용한 경우를 나타낸다.

유손실 모드는 무손실 모드에 어떤 수정 또는 변화를 줌으로써 손실이 발생하지만 그 대가로 압축률을 상당히 높일 수 있는 방식이다. JBIG2에서 유손실 모드는 여러 가지 방식이 존재한다. 첫 번째로 부호화 과정의 전처리로써 에지 완화와 고립된 화소들을 제거 시켜 주고 텍스트 영역에서는 심벌들의 정련 정보를 전송하는 방식, 두 번째로는 이런 전처리 후에 심벌들의 정련 정보조차도 보내지 않고 심벌 사전에 저장되어 있는 심벌 정보로만 심벌을 부호화하는 방식, 그리고 마지막으로 전처리도

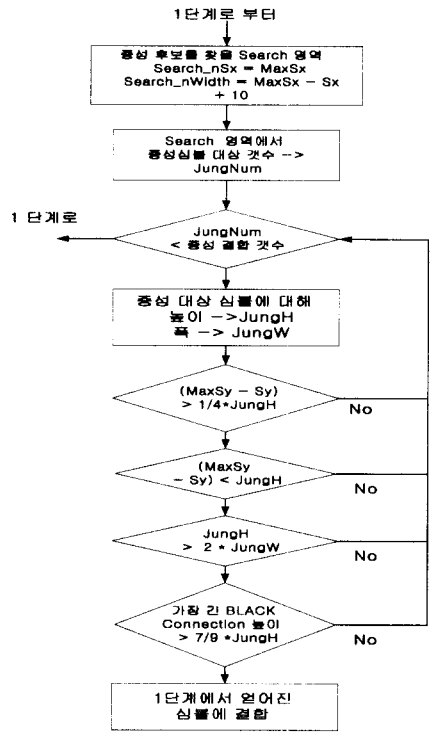
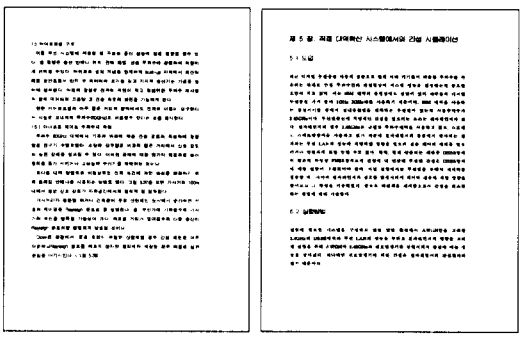


그림 4. 중성 모음을 결합하는 알고리즘



한글 문서 1　　한글 문서 2

그림 5. 실험에 사용된 문서

행하지 않고 텍스트 영역에서 심벌들의 정련 정보도 보내지 않는 유손실 방법 등이 존재하게 된다. 본 논문에서는 에지 완화와 고립화소의 제거 후 행하는 전처리 후에 텍스트 영역에서 심벌들의 정련 정보를 전송하는 첫 번째 방식에 대해서만 유손실 부호화를 실험했다. 다른 방식들은 훨씬 높은 압축률을 얻을 수 있지만 많은 실험과 심벌의 오매칭으로 인해 발생하는 의미자체의 부패를 막기 위한 더욱 정밀한 매칭 방법이 필요할 것이다.

1. 실험 결과

표5는 무손실 모드에서 비트맵을 허프만 부호화한 실험 결과를 보였다. M2는 M1에 비해 약 5~16(%) 정도 개선됨을 알 수 있었다. REFAGG 모드의 경우에는 M1에 비해 약 4~10(%) 정도 개선이 있었다. JBIG2 표준안에 의한 REFAGG 모드를 적용하는 경우에 비해 M2가 더 효율적임을 알 수 있었다.

표 5. 허프만 부호화에서의 M1, M2 and REFAGG의 비교(무손실모드, 단위; Byte)

문서 (DPI)	방법	심벌 사전	텍스트 영역			그 외	총 계	개선율 (%)
			인덱스	위치	정련			
문서1 (200)	M1	6,244	1,162	1,680	12,740	306	22,132	
	M2	7,511	622	932	9,118	259	18,442	16.67
	REFAGG	9,032	625	932	8,984	376	19,949	9.86
문서1 (300)	M1	8,765	1,186	1,936	17,055	298	29,240	
	M2	11,595	617	883	12,162	257	25,514	12.74
	REFAGG	12,666	617	883	12,093	373	26,632	8.92
문서1 (400)	M1	12,270	1,259	2,257	22,333	308	38,427	
	M2	15,848	620	939	16,690	260	34,357	10.60
	REFAGG	17,090	621	938	16,575	406	35,630	7.28
문서1 (600)	M1	21,882	1,320	2,785	30,212	340	56,539	
	M2	27,208	643	1,103	23,243	288	52,485	7.17
	REFAGG	27,801	644	1,103	23,313	435	53,296	5.74
문서2 (200)	M1	7,492	887	1,378	11,501	275	21,533	
	M2	7,631	579	901	9,949	256	19,316	10.30
	REFAGG	9,036	590	903	9,651	343	20,523	4.69
문서2 (300)	M1	10,729	908	1,537	16,100	273	29,547	
	M2	11,304	582	935	13,717	254	26,792	9.32
	REFAGG	12,470	587	937	13,444	341	27,779	5.98
문서2 (400)	M1	15,288	951	1,712	21,690	305	39,946	
	M2	16,031	590	957	19,068	257	36,903	7.62
	REFAGG	17,356	594	959	18,836	346	38,091	4.64
문서2 (600)	M1	26,664	1,005	2,217	29,156	293	59,355	
	M2	29,671	594	1,158	24,963	261	56,647	4.53
	REFAGG	30,206	597	1,159	24,749	382	57,093	3.81

표6은 무손실 모드에서 비트맵을 산술부호화한 실험 결과를 보였다. M2는 M1에 비해 약 3~10(%) 정도 개선됨을 알 수 있었다. REFAGG 모드의 경우에는 M1에 비해 약 2~6(%) 정도 개선이 있었다. 산술부호화를 적용한 경우가 허프만 부호화보다 개선 정도가 떨어지는 것을 알 수 있었다.

표7은 유손실 모드에서 비트맵을 허프만 부호화한 실험 결과를 보였다. M2는 M1에 비해 약 5~17(%) 정도 개선됨을 알 수 있었다. REFAGG 모드의 경우에는 M1에 비해 약 4~10(%) 정도 개선이 있었다. 무손실 모드에서와 거의 같은 이득을 얻을 수 있음을 알았다.

표8은 유손실 모드에서 비트맵을 산술부호화한

표 6. 산술부호화기에서 M1, M2 그리고 REFAGG의 비교 (무손실 모드, 단위: Byte)

문서 (DPI)	방법	심벌 사전	텍스트 영역		그 외	총 계	개선율 (%)
			인덱스 + 위치	정련			
문서1 (200)	M1	4,814	3,176	8,309	223	16,522	
	M2	6,025	1,485	7,105	195	14,810	10.36
	REFAGG	6,726	1,541	6,984	281	15,532	5.99
문서1 (300)	M1	6,588	3,559	12,130	223	22,500	
	M2	9,151	1,593	10,102	195	21,041	6.48
	REFAGG	9,486	1,632	10,051	281	21,450	4.67
문서1 (400)	M1	9,230	3,852	17,427	223	30,732	
	M2	12,632	1,670	14,681	195	29,178	5.06
	REFAGG	13,106	1,693	14,556	311	29,666	3.47
문서1 (600)	M1	15,026	4,146	25,434	247	44,853	
	M2	19,619	1,824	21,348	223	43,014	4.10
	REFAGG	20,005	1,867	21,376	336	43,584	2.83
문서2 (200)	M1	5,911	2,472	8,210	195	16,788	
	M2	6,172	1,429	7,913	195	15,709	6.43
	REFAGG	7,009	1,467	7,672	254	16,402	2.30
문서2 (300)	M1	8,251	2,745	12,563	195	23,754	
	M2	8,956	1,581	11,553	195	22,285	6.18
	REFAGG	9,590	1,624	11,344	254	22,812	3.97
문서2 (400)	M1	11,773	2,933	18,055	223	32,984	
	M2	12,710	1,658	16,963	195	31,526	4.42
	REFAGG	13,413	1,682	16,776	254	32,125	2.60
문서2 (600)	M1	18,356	3,170	25,649	223	47,398	
	M2	20,981	1,776	23,058	195	46,010	2.93
	REFAGG	21,396	1,798	22,876	283	46,353	2.20

표 7. 허프만 부호화에서의 M1, M2 and REFAGG의 비교(유손실모드, 단위; Byte)

문서 (DPI)	방법	심벌 사전	텍스트 영역			그외	총 계	개선율 (%)
			인덱스	위치	정련			
문서1 (200)	M1	5,803	1,156	1,660	11,808	304	20,731	
	M2	7,308	623	925	8,089	261	17,206	17.00
	REFAGG	8,818	626	925	8,009	376	18,754	9.53
문서1 (300)	M1	8,002	1,188	1,918	15,560	299	26,967	
	M2	10,654	615	863	10,858	260	23,250	13.78
	REFAGG	11,756	615	865	10,809	373	24,418	9.45
문서1 (400)	M1	11,405	1,270	2,248	20,884	311	36,118	
	M2	15,010	622	936	15,409	261	32,238	10.74
	REFAGG	16,216	623	936	15,362	404	33,541	7.13
문서1 (600)	M1	20,558	1,312	2,774	28,576	337	53,557	
	M2	25,515	638	1,095	22,129	287	49,664	7.27
	REFAGG	26,211	638	1,010	22,133	413	50,495	5.72
문서2 (200)	M1	7,137	872	1,364	10,809	275	20,457	
	M2	7,380	580	901	9,172	256	18,289	10.60
	REFAGG	8,748	589	906	8,875	343	19,461	4.87
문서2 (300)	M1	10,158	895	1,511	14,912	271	27,747	
	M2	10,688	577	915	12,620	254	25,054	9.71
	REFAGG	11,719	580	920	12,395	340	25,954	6.46
문서2 (400)	M1	14,599	947	1,689	20,423	305	37,963	
	M2	15,603	587	948	17,716	258	35,112	7.51
	REFAGG	16,739	590	949	17,535	345	36,158	4.75
문서2 (600)	M1	25,563	996	2,210	28,136	313	57,218	
	M2	28,068	592	1,151	24,314	262	54,387	4.95
	REFAGG	28,904	594	1,152	24,050	381	55,081	3.73

실험 결과를 보였다. M2는 M1에 비해 약 3~10(%) 정도 개선됨을 알 수 있었다. REFAGG 모드의 경우에는 M1에 비해 약 2~6(%) 정도 개선이 있었다. 위의 표7의 결과와 같이 무손실 모드에서와 거의 같은 이득을 얻을 수 있음을 알 수 있었다.

실험 결과를 분석해 보면 M2의 경우는 M1에 비해 심벌의 결합으로 인한 매칭의 감소로 인해 심벌 사전의 크기는 증가되지만 인덱스와 위치정보 및 정련 정보가 대폭 감소됨을 알 수 있다. REFAGG 모드의 경우는 정련 정보는 M2에 비해 더욱 감소하지만 심벌 사전이 상대적으로 더 많이 증가함을 알 수 있다. 그러나, 정련 정보와 심벌 사전의 크기는 상대적이고 trade-off가 있음을 알 수 있다.

표 8. 산술부호화에서의 M1, M2 그리고 REFAGG의 비교(유손실 모드)

문서 (DPI)	방법	심벌 사전	텍스트 영역		그 외	총 계	개선율 (%)
			인덱스 + 위치	정련			
문서1 (200)	M1	4,424	3,169	7,421	223	15,237	
	M2	5,814	1,487	6,193	195	13,689	10.16
	REFAGG	6,429	1,503	6,147	281	14,360	5.75
문서1 (300)	M1	5,870	3,543	10,702	223	20,338	
	M2	8,235	1,597	8,791	195	18,818	7.47
	REFAGG	8,593	1,621	8,761	281	19,256	5.32
문서1 (400)	M1	8,395	3,846	16,020	223	28,484	
	M2	11,727	1,678	13,413	195	27,013	5.16
	REFAGG	12,215	1,714	13,376	311	27,616	3.04
문서1 (600)	M1	13,750	4,100	23,822	247	41,919	
	M2	18,008	1,819	20,171	223	40,221	4.05
	REFAGG	18,441	1,837	20,222	311	40,811	2.64
문서2 (200)	M1	5,529	2,446	7,564	195	15,734	
	M2	5,906	1,419	7,207	195	14,727	6.40
	REFAGG	6,641	1,461	6,975	254	15,331	2.56
문서2 (300)	M1	7,649	2,742	11,357	195	21,943	
	M2	8,301	1,566	10,508	195	20,570	6.26
	REFAGG	8,828	1,591	10,313	254	20,986	4.36
문서2 (400)	M1	11,062	2,913	16,885	223	31,083	
	M2	12,200	1,661	15,664	195	29,720	4.38
	REFAGG	12,750	1,689	15,495	254	30,188	2.88
문서2 (600)	M1	17,220	3,173	24,608	223	45,224	
	M2	19,494	1,753	22,420	195	43,862	3.01
	REFAGG	20,068	1,809	22,137	283	44,297	2.05

표9는 ITU-T에서 제공한 로마자 표준 문서에 대하여 적용한 실험 결과를 보였다. 한글에 적용되는 알고리즘이지만 표준 문서에 대해서도 이상 없이 동작함을 보였다, 단지, f04문서의 경우 해상도가 높으면 성능이 열화 됨을 알 수 있었다. 결과적으로 본 알고리즘은 한글에 대하여 더욱 효율적인 방법임을 보여 준다.

표 9. 표준 로마자 문서 적용 실험 결과(무손실 모드, 단위 : Byte)

방법 / 영문서(dpi)	부호화 방법	M1	M2	개선율 (%)
f01 (200)	허프만 부호화	14,824	14,387	2.95
	산술 부호화	10,172	10,032	1.38
f01 (300)	허프만 부호화	19,693	19,316	1.90
	산술 부호화	14,708	14,624	0.57
f01 (400)	허프만 부호화	25,451	25,130	1.26
	산술 부호화	19,773	19,752	0.11
f04 (200)	허프만 부호화	55,920	53,280	0.05
	산술 부호화	36,958	37,385	-1.16
f04 (300)	허프만 부호화	71,927	71,792	0.19
	산술 부호화	52,516	54,749	-4.25
f04 (400)	허프만 부호화	93,682	94,760	-1.15
	산술 부호화	72,868	75,612	-3.77

2. 고 찰

문서의 해상도가 증가하면 압축의 개선율이 감소함을 볼 수 있다. 이는 해상도의 증가로 심벌의 크기가 커졌고 매칭시 ±2정도의 크기 차가 있는 심벌들에 대해서만 매칭을 시도했기 때문이다.

REFAGG방법은 M2에 비해 압축률 측면에서 문서 1인 경우에는 1.3~6.8(%), 문서 2의 경우에는 0.7~5.6(%) 떨어짐을 알 수 있었다. REF- AGG 방법이 압축률이 떨어지는 이유는 영문의 경우 결합된 심벌 "the"의 "t", "h" 그리고 "e"등은 다시 문서의 다른 심벌 "t", "h" 등과 매칭을 시도하게 되지만 한글의 경우 "문"자의 "ㅁ", "ㅜ" 그리고 "ㄴ"등은 이미 결합 문자가 된 심벌과 매칭을 시도하게 되지만 심벌의 크기가 다르기 때문에 매칭을 전혀 하지 않게 된다. 때문에 "ㅁ", "ㅜ" 그리고 "ㄴ"은 매칭과 전혀 관계없는 잉여 성분이 될 수밖에 없다. 그러나 만약 "문"과 "자"가 단독이 아닌 영문의 "the"처럼 결합된 심벌인 "문자"의 빈도가 많이 발생할 때는 영문과 마찬가지 방식으로 "문"과 "자"를 REFAGG 부호화 방식으로 부호화해도 효과가 있을 것이다.

본 방식의 단점은 부호화 시간이 더 소요되는 점이다. M2 방법은 전처리에 시간이 더 소요되므로 M1에 비해 약 12~30(%) 정도의 시간이 더 필요한 단점이 있다. 한편 REFAGG 방법을 적용한 경우는 M2에 비해 4~20(%)정도의 더 많은 시간이 필요했다.

또한 심벌의 결합 시에 한글에 영문이 혼합되어 있는 경우라든지 숫자와 관련된 부분에서 원하지 않는 결합이 이루어 졌음을 볼 수 있었다. 문서 2에서 "AW"와 "GH"의 경우 제안한 알고리즘의 1단

계에 의해서 결합된 문자가 되었음을 볼 수 있었다. 그리고, 본 논문은 문서가 다단으로 나누어지지 않은 경우만을 생각했다. 다단으로 나누어졌다든지 여러 가지 다양한 패턴에 대해서는 영역 분할(segmentation) 방법을 적용함으로써 본 논문에서 제안한 방식이 이용 가능할 것이다.

V. 결 론

본 논문에서는 한글에 JBIG2를 적용 시에 효율적인 압축 방법을 제안했다. 한글은 로마계열 문자인 영문과 달리 심벌 추출을 하면 자소 단위 심벌로 쪼개질 수밖에 없다. 이것은 심벌의 위치 정보를 전송하기 위해 더 많은 비용을 지불해야 함을 의미한다. 따라서 본 논문은 이런 위치 정보를 감소시키기 위해 자소 단위 심벌을 영문의 알파벳처럼 다룰 수 있도록 하기 위해 의미 있는 문자단위 심벌로 결합하는 알고리즘을 제안했다.

제안한 알고리즘은 산술부호화 보다는 허프만 부호화에서 더 많은 개선을 얻을 수 있었다. 무손실의 경우 허프만 부호화를 적용 시 제안한 알고리즘을 적용한 방법이 적용하지 않은 방법에 비해 고해상도에서는 4.5~10.6(%), 저해상도에서는 9.3~16.7(%) 정도의 향상된 압축률을 얻을 수 있었다. 산술부호화를 적용할 경우에는 허프만 부호화기에 비해 개선율은 떨어졌지만 고해상도에서는 2.9~5.1(%), 저해상도에서는 6.2~10.4(%) 정도의 개선을 얻을 수 있었다. 유손실 모드에서는 4.0~10.6(%) 정도 압축률의 증가를 얻을 수 있었다.

제안한 알고리즘을 적용할 경우 심벌의 결합에 필요한 부호화 시간이 12~30(%) 정도 더 필요함을 확인하였다. 제안한 알고리즘이 압축 성능 면에서 JBIG2의 REFAGG보다 우수함을 알 수 있었고, REFAGG 부호화는 부호화 시간 측면에서도 제안한 알고리즘에 비해 4~20(%) 정도가 더 소요됨을 알 수 있었다.

또한 본 논문에서 다루지 않은 다단 문서에서도 세그먼트 기법을 이용한 영역 분할에 의해서 제안한 알고리즘을 사용할 수 있을 것이다.

참 고 문 헌

[1] *ISO/IEC JTC1/SC29/WG1 N1359R,* JBIG2 Final Draft International Standards, Dec. 1999.

[2] CCITT Draft Rec. T.4, *Standardization of Group 3 Facsimile Apparatus For Document Transmission*, 1979.

[3] CCITT Rec. T.6, *Facsimile Coding Schemes and Coding Control Functions for Group4 Facsimile Apparatus,* 1988.

[4] ITU-T Rec. T.82, *Information Technology -Coded representation of Picture and Audio Information - Progressive Bi-Level Image Compression,* March 1993.

[5] W. Rucklidge, "Xerox Proposal for JBIG2 Coding," *ISO/IEC/JTC1/SC29/WG1N339* June 1996.

[6] R. Arps and C. Constantinescu, "Blocked Symbol Type(BST) Compression Coding Method for JBIG2," *ISO/IEC JTC1/ SC29/WG1 N340Rev,* June 1996.

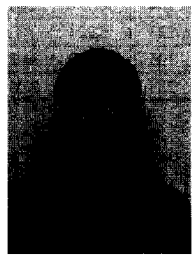
[7] O. Johnsen, J. Segen, and G. L. Cash, "Coding of Two-Level Pictures by Pattern Matching and Substitution," *Bell System Technical Journal,* vol. 62, no. 8, Oct. 1983.

[8] I. H. Witten, T. C. Bell, H. Emberson, S. Inglis, and A. Moffat, "Textual Image Compression : Two-Stage Lossy/Lossless Encoding of Textual Images," *Proc. of The IEEE,* vol. 82, no. 6, pp.878-888, June 1994.

[9] P. G. Howard, "Lossless and Lossy Compression of Text Images by Soft Pattern Patching," *ISO JTC1/SC29/ WG1 N205*, 1995.

[10] 김영태, 고형화, "패턴매칭에 의한 2진 한글 문서의 유·무손실 압축에 관한 연구," *한국 통신학회 논문지*, 제22권, 제4호, pp. 726-736, 1997.

강 병 택 (Byung-taek Kang) 정회원

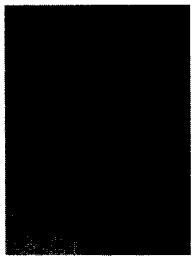


1998년 2월 : 서울산업대학교 전자공학과 졸업

2000년 2월 : 광운대학교 전자통신공학과 석사

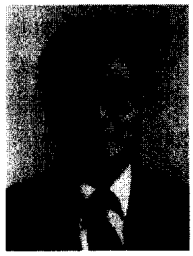
2000년2월~현재 : (주)삼성전기 시스템연구실 연구원

김 현 민(Hyun-min Kim) 정회원



1993년 2월 : 독학 이학사
1996년 8월 : 광운대학교 전자통신공학과 석사
1999년 8월 : 광운대학교 전자통신공학과 박사과정수료
1999년~현재 : 광운대학교 전자통신공학과 연구과정
<주관심 분야> 영상압축, 신호처리, 통신시스템

고 형 화(Hyung-hwa Ko) 정회원



1979년 2월 : 서울대학교 전자공학과 졸업
1982년 2월 : 서울대학교 전자공학과 석사
1989년 2월 : 서울대학교 전자공학과 박사
1979년~1980년 : (주)금성사 중앙연구소
1985년~1987년 : 광운대학교 전자통신공학과 전임강사
1987년~1991년 : 광운대학교 전자통신공학과 조교수
1991년~1996년 : 광운대학교 전자통신공학과 부교수
1996년~현재 : 광운대학교 전자공학부 정교수
1998년 8월~1999년 8월 : UCSD 객원교수
<주관심 분야> 2진영상 압축, 동영상 압축 및 전송, Wavelet-based 영상처리, 디지틀 통신