모바일 게임을 위한 개선된 무손실 이미지 압축

김 세 웅[†] · 조 병 호[††]

요 약

본 논문에서는 모바일 게임의 전체 용량 중 상당 부분을 차지하는 이미지를 무손실로 압축하기 위한 방법을 제안하였다. 이미지의 압축률을 높이기 위해 실제로 압축을 수행하기 전에 전처리 과정에서 이미지를 재구성 한 후 RFC-1951에 정의된 Deflate 알고리즘으로 압축하였다. 전처리 과정에서는 이미지의 정보를 바탕으로 사전 기반 부호화의 특징인 사전의 크기를 얻고, 픽셀 패킹과 DPCM 예측 기법을 사용하여 이미지를 재구성하는 방법을 사용하여 일반적인 방법으로 압축할 때 보다 압축률을 향상시켰다. 제안된 압축 방법을 다양한 모바일 게임 이미지에 적용하여 압축률을 실험한 결과 기존 모바일 이미지 포맷에 비해 약 9.7%의 압축률이 향상됨을 보였다.

키워드 : 모바일 게임, 무손실 이미지 압축, 모바일 플랫폼, BREW

An Improvement of Lossless Image Compression for Mobile Game

Se-Woong Kim[†] · Byung-Ho Jo[††]

ABSTRACT

In this paper, the method to make lossless image compression that holds considerable part of total volume of mobile game has been proposed. To increase the compression rate, we compress the image by Deflate algorithm defined in RFC 1951 after reorganize it at preprocessing stage before conducting actual compression. At the stage of preprocessing, we obtained the size of a dictionary based on the information of image which is the feature of Dictionary-Based Coding, and increased the better compression rate than compressing in a general manner using in a way of restructuring image by pixel packing method and DPCM prediction technique. It has shown that the method increased 9.7% of compression rate compare with existing mobile image format, after conducting the test of compression rate applying the suggested compression method into various mobile games.

Key Words : Mobile Game, Lossless Image Compression, Mobile Platform, BREW

1. 서 론

유비쿼터스로 대변되는 컴퓨터와 통신의 융합으로 인해 다양한 멀티미디어 기기와 컨텐츠가 등장하였다. 현재 3천만명이 넘는 인구가 휴대전화 단말기를 사용하고 있으며, 그 중 기본적인 통화와 문자메시지를 제외하면 수많은 모바일 컨텐츠 중 단연코 게임을 빼놓을 수 없다[1]. 이러한 모바일 게임은 최초 흑백 LCD에서의 WAP(Wireless Application Protocol) 게임을 시작으로 하여 최근에는 트루컬러(TrueColor)와 64Poly 이상의 사운드를 지원하는 3D 게임에 이르기까지 다양하게 발전하였다. 그러나 하드웨어의 발전은 빠르게 이루어졌지만 대다수의 모바일 게임은 일반적으로 보급되어 있는 휴대전화 단말기에 맞춰서 개발되므로 높아져만 가는 사용자의 눈높이를 맞추기에는 메모리나 단말기의 동작 속도 등의 제한사항이 따르지 못하는 문제로 인해 각종 데이터에 대한 압축 방법이 필요하였다.

이중 모바일 게임에서 상당량을 차지하는 이미지를 압축하기 위해 초기에는 기초적인 RLE(Run Length Encoding)[2] 방법을 사용하여 이미지를 압축하였으나 복잡한 이미지에서는 압축률이 낮으므로 차츰 각종 2D 기반의 압축 방법[3]들의 사용, 모바일 플랫폼인 BREW(Binary Runtime Environment for Wireless)[4]나 J2ME기반의 MIDP(Mobile Information Device Profile)[5] 등에서 PNG를 사용하는 등 기존 포맷의 사용이나 기존 포맷을 개선[6] 하는 등의 방법을 사용하였다. 이외에도 모바일 플랫폼에서 기존 포맷에 비해 보다 압축률이 높은 모바일 전용 압축 솔루션[7-10]들이 개발되어 사용되었다.

그러나 점차 높아져만 가는 사용자의 요구를 충족시키기 위해서 기존의 중복되는 패턴 형식의 이미지에서 탈피하여

† 정 회 원 : 서울산업대학교 컴퓨터공학과 석사과정
†† 종신회원 : 서울산업대학교 컴퓨터공학과 교수
논문접수 : 2006년 2월 15일, 심사완료 : 2006년 4월 13일

3D 렌더링을 이용한 2D 이미지들이 사용되기 시작하였고, 광원효과 등을 위해 알파블랜딩(alpha blending)등의 특수효과를 사용하는 게임들이 증가하면서 이를 처리하기 위해 내부적으로 이미지의 비트맵(bitmap)을 자유롭게 다룰 수 있는 기법들이 사용되기 시작하였다. 또한, 휴대전화 단말기의 가능한 색 표현 수가 증가하면서 점차적으로 65000색 이상을 사용하는 게임들도 증가하고 있다. 이로 인해 플랫폼 의존적인 압축 방식을 벗어나면서 기존 압축 방법에 비해 좀 더 효율이 좋고 24비트 등 다양한 이미지의 압축 방법에 관한 연구가 필요하게 되었다.

본 논문에서는 GIF, PNG등의 기존 이미지 포맷이나 상용 모바일 이미지 솔루션에 비해 다양한 이미지에서 보다 압축률을 향상시킬 수 있는 방법을 제시하였다. 모바일 게임에 사용되는 이미지의 압축 효율을 높이기 위해 전처리(preprocessing) 과정에서 이미지의 정보를 바탕으로 LZ77 등의 사전 기반 부호화(Dictionary Based Coding)[11]에서의 특징인 사전의 크기를 설정하고 이미지를 픽셀 패킹(pixel packing)과 DPCM(Differential Pulse Code Modulation)[12] 단계를 거친 후 RFC-1951에 정의된 Deflate 알고리즘[13]으로 압축하였다. 이로 인해 단순한 방법으로 압축할 때 보다 압축률을 향상시켰다.

2. 모바일 게임 이미지 압축의 개선

2.1 모바일 게임 이미지 압축시 고려사항

휴대전화 단말기를 기반으로 하는 모바일 환경은 작은 해상도와 한정된 메모리 공간, 적은 어플리케이션의 크기로 제한되는 특징을 가지고 있다. 이러한 모바일 환경은 게임에 그대로 반영이 되므로 일반적으로 개발되는 모바일 게임의 환경은 <표 1>과 같이 나타낼 수 있다.

〈표 1〉 일반적인 모바일 게임의 특징

항목	특 징
게임 화면 크기	120×117~240×296
용량	250KB~300KB
사용 색 수	256색 사용
이미지 특징	픽셀 아트 방식의 그래픽 이미지

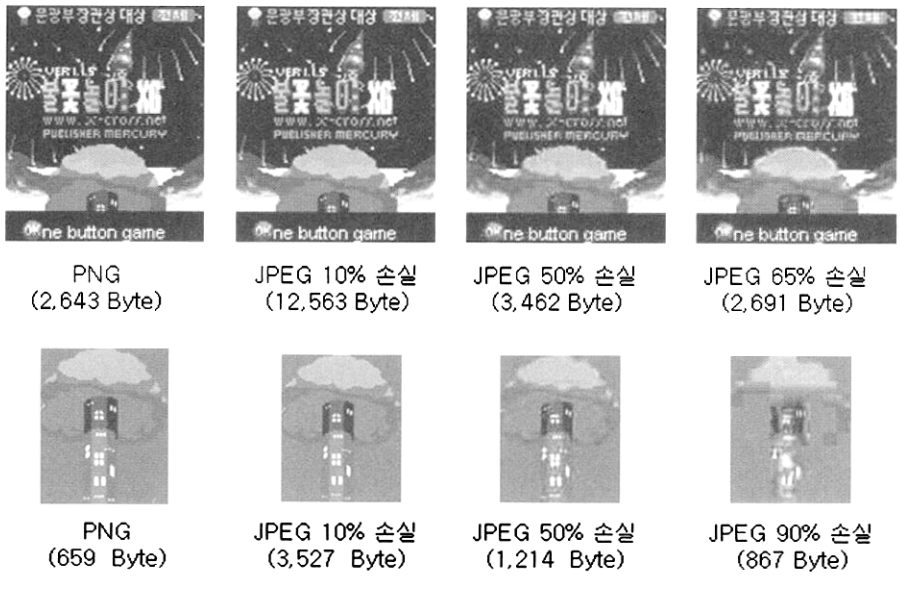
최근에는 다양한 컬러를 지원하는 휴대전화 단말기들이 등장했지만 기존 단말기를 사용하는 사용자층도 포함해야 하고 각 통신사의 용량제한 등의 정책에도 부응해야 하므로, 다양한 이미지나 데이터의 압축률을 높여 용량을 최소화해야 한다.

이렇듯 일반적인 모바일 게임은 PC 게임 등에 비해 적은 데이터로 고품질의 다양한 이미지를 표현해야 하므로 모바일 게임에 사용되는 이미지를 압축하기 위해서는 다음과 같은 몇 가지의 고려사항이 있다.

① 픽셀 단위로 그려진 이미지를 위한 무손실 압축
② 모바일 플랫폼을 위한 적은 메모리의 사용
③ 모바일 게임을 위한 빠른 압축 복원 속도
④ 특정 플랫폼에 종속되지 않는 일반적인 압축 알고리즘의 사용
⑤ 이미지 비트맵 데이터의 자유로운 조작
⑥ 점차 사용이 확대되는 24비트 이미지의 압축도 고려

모바일 게임의 경우 픽셀 단위로 그려지는 이미지가 대부분이므로 손실 압축을 사용하여 이미지를 압축한다면 이미지의 용량은 줄어들겠지만 사진의 이미지와 비교하면 품질은 더욱 저하된다.

(그림 1)에서 모바일 게임의 이미지를 손실 압축 하였을



(그림 1) 모바일 게임 이미지의 손실 압축 시 품질 저하

때 품질이 저하되는 것을 볼 수 있다. 이미지의 비교는 최근 모바일 플랫폼에서 많이 사용하고 있는 PNG와 대표적인 손실 압축 방식인 JPEG를 대상으로 하였다. 일반적인 실사 사진의 경우 10% 정도의 손실 압축으로도 큰 차이를 느끼지 못하지만 모바일 게임 같은 픽셀 단위의 그래픽 이미지에서는 10% 정도의 손실로도 전체적으로 색감이 흐려지는 등의 품질 저하를 느낄 수 있다. 특히 PNG의 사이즈와 비슷한 크기를 얻는 손실률에서는 이미지의 손실이 확연하게 드러난다.

이렇듯 손실 압축을 사용할 경우 게임에 사용되는 이미지는 인접 픽셀의 격차가 크고 전체적인 크기가 작으므로 조그마한 손실로도 그래픽 디자이너가 의도하지 않은 품질 저하가 발생하며 동시에 무손실 압축에 비해 용량이 증가하는 등의 비효율적인 면으로 인해 모바일 게임 등의 픽셀 아트 방식의 그래픽 이미지에서는 무손실 압축이 가장 효과적이다.

압축 알고리즘의 선택은 다양한 사전 기반 부호화 알고리즘 중에서 PNG 등의 이미지 포맷에 사용하고 있어 모바일 플랫폼에도 사용 가능하며 RFC-1951에 공개되어 있는 Deflate 알고리즘을 주된 압축 알고리즘으로 사용하였다.

2.2 모바일 게임 이미지 압축의 개선 방향

일반적으로 모바일 게임은 제한된 환경과 휴대전화 단말기의 색 표현 제한으로 인해 주로 8비트 256색을 사용하여 이미지를 표현하는데, 이때 사용되는 팔레트(palette)는 1~3개를 주로 사용하게 된다. 그러므로 중복되어 사용되는 팔레트는 이미지에서 따로 분리하여 실제 비트맵만을 압축하는 방식으로 진행한다.

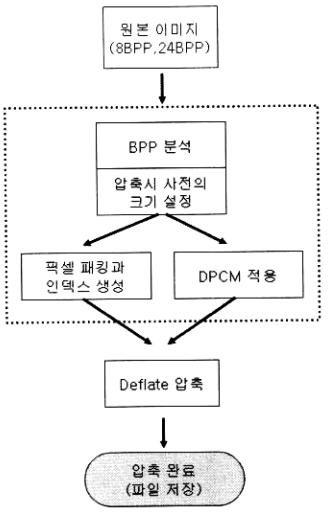
이때 모바일 게임 이미지의 압축 효율을 높이기 위해 몇 가지의 변환 과정을 추가하면 단순 압축에 비해 압축률을 높일 수 있다. 이에 본 논문에서는 전처리 과정을 통해 1차적으로 데이터의 분석과 변환 단계를 거친 후 실제로 압축을 수행하는 방법을 통해 압축률을 향상시킬 수 있는 방법을 제안한다.

(그림 2)에 본 논문에서 제안한 모바일 게임 이미지의 압축 과정을 요약하였다. 제안된 압축 방법은 입력된 비트맵 데이터를 분석하여 압축시 사전의 크기를 설정하고 이후 픽셀 패킹 또는 DPCM 기법을 적용하는 전처리 과정과 실제로 압축을 수행하는 Deflate 압축부로 나누어진다.

점선 부분이 전처리 과정을 뜻한다. 전처리 과정은 다시 3번의 세부 단계를 거치게 되는데 이것은 BPP(Bit Per Pixel) 분석, 압축시 사전의 크기 설정, 데이터 재구성으로 이루어진다. 데이터 재구성 단계는 8비트 컬러 이미지, 그레이 이미지와 24비트 이미지에 따라서 픽셀 패킹 또는 DPCM 적용 단계로 또 다시 나누어진다.

이러한 전처리 과정을 수행한 후의 데이터는 Deflate 알고리즘을 사용하여 실제로 압축하게 되며 이미지의 정보를 담은 헤더와 압축된 데이터를 포함하여 파일로 저장하는 것을 끝으로 제안한 모바일 게임 이미지 압축의 전 과정을 마치게 된다.

압축된 이미지를 모바일 플랫폼에서 실제로 사용하기 위해 복원할 때는 압축 과정을 역순으로 진행하게 된다. 즉, 압축된 이미지를 Deflate 복원 알고리즘으로 1차로 복원한 다음 후처리(postprocessing) 과정을 통해 실제 이미지의 비트맵 데이터를 얻게 된다. 후처리 과정에서는 헤더에 기록된 정보를 바탕으로 픽셀 복원인지 DPCM 복원인지를 결정한다. 마지막으로 복원된 비트맵 데이터를 모바일 플랫폼에서 사용하기 위해 따로 저장되었던 팔레트를 결합하고 각 플랫폼에 맞는 내부 형식으로 변환하여 최종 이미지를 완성한다.



(그림 2) 제안한 모바일 이미지 압축 과정

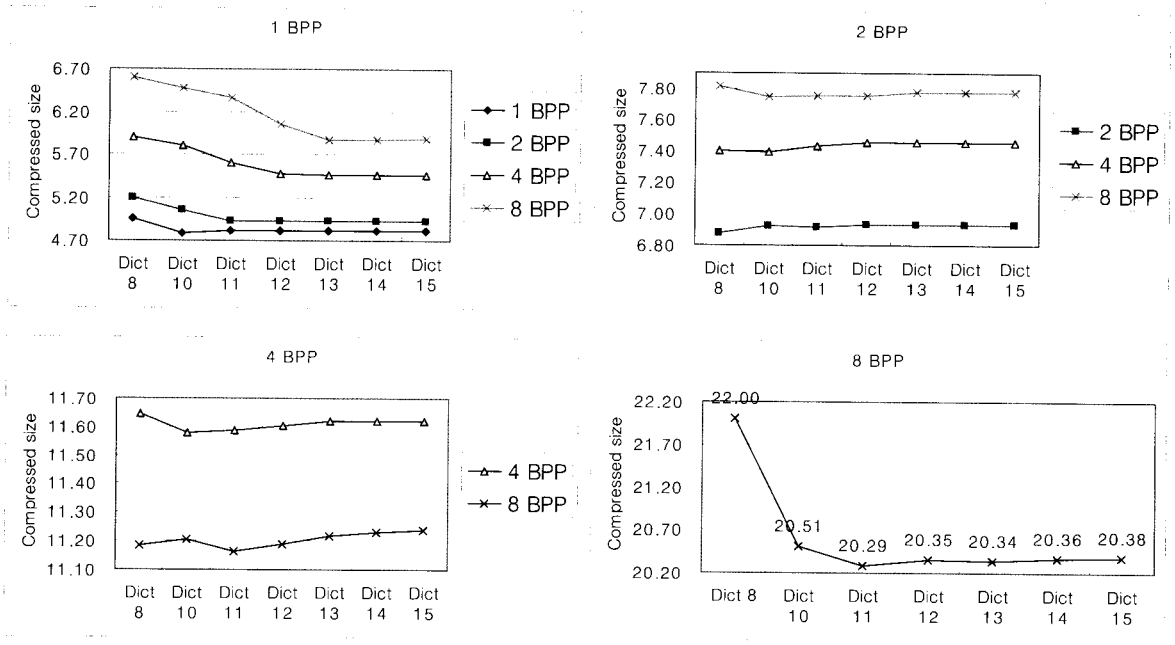
2.3 이미지의 전처리 과정

2.3.1 BPP에 따른 압축 사전의 크기 설정

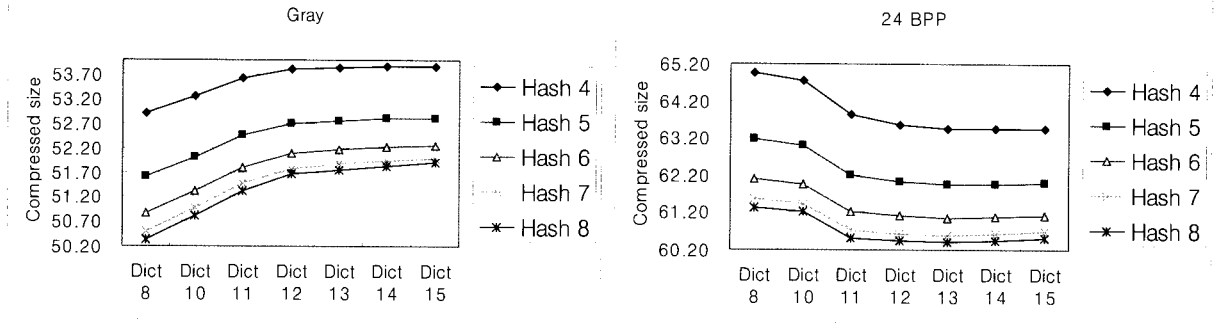
본 논문에서 사용하는 Deflate 알고리즘 등의 사전 기반 부호화 방식에서는 압축시에 사용하는 사전의 크기가 압축률에 영향을 준다. 때문에 데이터에 따라 이를 적절하게 증감시켜서 압축하게 된다면 고정된 크기에 비해 압축 효율을 높일 수 있다.

이를 이용하여 본 논문에서는 전처리 과정중 압축 사전의 크기 설정 단계에서 각 BPP에 대해서 최적의 압축률을 얻을 수 있는 사전과 해쉬 테이블의 크기를 결정하기 위해 이미지의 BPP와 이미지의 크기, 그리고 사전의 크기에 대한 압축률을 시뮬레이션을 통하여 실험하였으며 이때 생성된 데이터를 실제 압축에서 적용하였다. 이때 Deflate 알고리즘에서 사용하는 해쉬의 크기는 BPP 값이 높아짐에 따라 약간씩 증가하는 것이 압축률 향상에 도움이 되었다. 압축 사전의 크기를 설정하기 위한 이미지의 BPP는 256바이트의 간단한 해쉬 테이블을 사용하여 찾을 수 있으며 이때 사용한 해쉬 테이블은 픽셀 패킹에서 재사용한다.

각 BPP와 사전의 크기에 대한 시뮬레이션 결과를 (그림 3)와 (그림 4)에 나타냈다. 사전 기반 부호화에서는 일반적으로 사전의 크기가 클수록 압축률이 증가하지만 1BPP나 2BPP, 그레이 이미지의 경우처럼 데이터가 복잡하지 않을 경우는 사전의 크기가 작은 것이 압축률 향상에 도움이 된



(그림 3) 8BPP 컬러 이미지의 사전 크기 분석



(그림 4) DPCM 적용 이미지의 사전 크기 분석

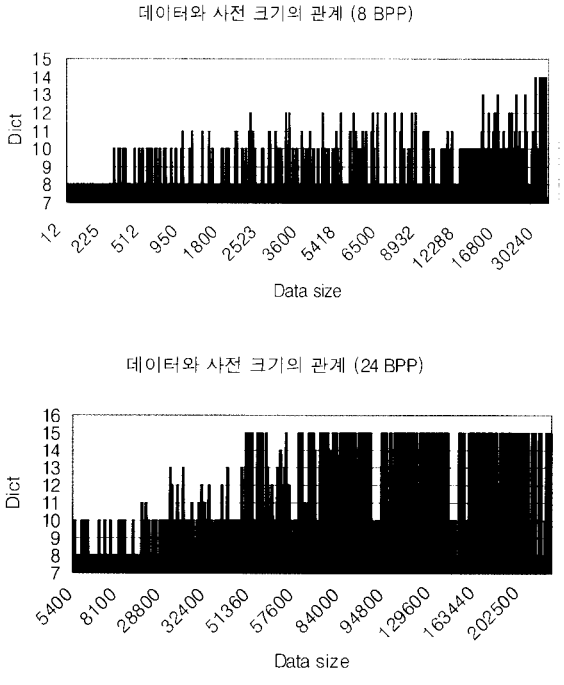
다는 것을 알 수 있다. 반대로 24BPP 이미지의 경우는 사전의 크기와 압축률이 대체로 비례하는 것을 볼 수 있다.

이를 좀 더 분석해 보면 1BPP, 2BPP의 경우는 픽셀 패킹으로 인해 4BPP 이상의 BPP를 적용할 때 보다 압축률이 향상되었고, 4BPP 는 8BPP를 적용하는 것이 압축률 향상에 도움이 되는 것으로 나타났다. 마지막으로 8BPP 이미지는 일반적으로 사전의 크기에 따라 압축률이 높아졌지만 모바일 게임 이미지의 특성이라 할 수 있는 작은 사이즈로 인해 15비트의 사전 크기일 때는 압축률이 소폭 감소하였다.

DPCM을 적용한 그레이와 24BPP 이미지에서는 전체적으로 해쉬 테이블의 크기가 클수록 압축률이 향상되었으며, 그레이 이미지와 24BPP 이미지의 압축 사전의 크기는 서로 상반된 결과를 보였다. 또한, 24BPP 이미지에서는 14비트 이상의 사전의 크기일 때 압축률이 오히려 하락하는 것을 볼 수 있다. 이는 모바일 게임에서 사용하는 작은 사이즈의 24BPP 이미지의 경우이므로 PC등에서 사용하는 이미지에서는 사전의 크기가 큰 것이 압축률 향상에 도움이 된다.

시뮬레이션에서 사용한 이미지는 전체 모바일 게임을 대상으로 한 것은 아니지만 8BPP 컬러 이미지에서 압축률 향상에는 일반적으로 BPP가 낮을 때는 사전의 크기가 작은 것이 도움이 되었고 BPP가 높을 때는 사전의 크기가 큰 것이 도움이 되었다. 또한, 데이터의 크기에 따라서도 사전의 크기를 다르게 설정하면 보다 압축률을 높일 수 있다. (그림 5)를 보면 대체적으로 데이터의 크기에 비례하여 사전의 크기가 커지는 것을 볼 수 있는데, 8BPP 이미지의 경우 대체적으로 사전의 크기가 13비트 이하임을 알 수 있으며, 24BPP 이미지에서는 이미지 자체의 BPP가 크기 때문에 8BPP와 같은 해상도를 가지는 이미지에서도 사전의 크기가 커지는 것을 볼 수 있다. 이렇게 데이터의 크기에 따른 사전의 크기 또한 각 BPP에 따라 구분하여 적용하면 보다 압축률을 향상시킬 수 있다.

시뮬레이션을 통해 분석한 자료를 바탕으로 이미지의 BPP와 데이터의 크기에 대한 각 사전의 크기를 기본값으로 설정하여 그 값을 <표 2>에 나타내었다. 256 그레이 이미지의 경우에는 512×512 사이즈에서도 최적 사전의 크기가 8을 크게 벗어나지 않았으므로 모바일에서의 사용은 사전 크기 8로도 충분하다.



(그림 5) 데이터와 사전 크기의 관계

〈표 2〉 이미지의 사전 크기 설정

기준색	BPP	데이터 크기 (KByte)	사전 크기 (2n Byte)	해쉬 크기 (2n Byte)
2	1	20 초과	13	7
		20	12	7
		10	11	7
		4	8	5
4	2	1 초과	11	7
		1	8	5
16	8	1 초과	11	7
		1	10	5
256	8	20 초과	14	7
		20	13	7
		10	12	7
		1	11	7
256 Gray	8	256	8	8
224	24	128 초과	15	8
		128	13	8
		64	10	8

2.3.2 픽셀 패킹 및 인덱스 생성

픽셀 패킹이란 8BPP 컬러 이미지에서 수행하는 단계로 4BPP 이하의 원본 비트맵 데이터를 실제 사용하는 비트수로 재구성한 후 1바이트로 묶는 것을 말한다.

우선 이미지에서 사용된 색이 16색 이하라 하더라도 실제로는 256색 중에서 선택된 색이므로 0~255의 픽셀 값을 0~15의 값이 되도록 재구성하는 과정을 거쳐 픽셀을 4비트 이하의 값으로 패킹하게 된다.

이때 (그림 3)에서 나타난 4BPP 이하의 결과를 보게 되면 1BPP와 2BPP 에서는 픽셀 패킹을 하는 것이 그렇지 않을 때 보다 압축률이 향상되지만 4BPP 에서는 픽셀 패킹을 하지 않는 것이 압축 효율이 좋은 것을 알 수 있으므로 픽셀 패킹은 2BPP 이하에서만 적용한다. 이 결과로 픽셀 패킹이 적은 수의 색을 사용하는 이미지에서 압축률 향상에 도움이 된다는 것을 알 수 있다.

2.3.3 DPCM 적용

DPCM 적용은 8BPP 그레이 이미지와 24BPP 이미지에서만 수행하는 단계로 예측 부호화 방식 중 DPCM을 적용하여 압축을 수행한다.

DPCM을 적용하면 인접한 픽셀의 차를 이용하여 이미지의 복잡도를 낮춰서 압축률을 향상시킬 수 있다. 이때 픽셀 예측은 스캔라인 단위로 현재 픽셀과 왼쪽, 위쪽 데이터의 차를 이용한다. 이때 사용되는 계산식은 식 (1)과 (2)를 사용한다. 이때 $pixel(cur)$는 현재 위치의 픽셀이고, $pixel(left)$와 $pixel(up)$은 왼쪽과 위쪽의 픽셀이며, $pd1_i$와 $pd2_i$는 생성되는 예측값이다.

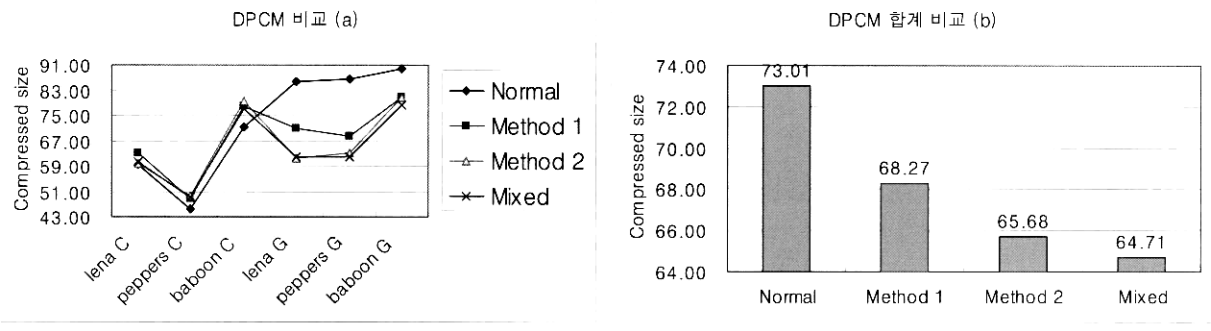
$$\text{(Method 1)}\quad pd1_i = pixel(left) - pixel(cur) \qquad \text{식 (1)}$$

$$\text{(Method 2)}\quad pd2_i = pixel(up) - pixel(cur) \qquad \text{식 (2)}$$

(그림 6)의 (a)에서는 각각의 이미지에 대해 각 방법의 예측 계산을 하여 이미지를 압축한 결과를 비교하였고, (b)에서는 총 합계를 좀 더 자세히 비교하여 제시하였다. 결과를 보면 일반적으로 Method 2를 사용했을 때 압축률이 향상되는 것을 알 수 있다. 그러나 데이터에 따라 Method 1의 압축률이 좋은 경우도 있으므로 이 두 가지의 방법을 혼합하여 사용한다면 좀 더 압축률을 높일 수 있다. 혼합할 때는 우선 스캔라인 단위로 각 방법을 사용하여 예측값을 생성한다. 이후에 H.262, H.263등의 동영상 압축 표준에서 ME(Motion Estimation)의 블록매칭(Block Matching)에서도 사용하는 SAD(Sum of Absolute Difference)의 값을 각 예측방법에 적용한 후 가장 작은 값을 가지는 방법을 택한다 [14, 15]. 이때 사용한 SAD의 값은 식(3)을 사용하여 구한다. 여기서 pd_i는 각 픽셀의 예측값이고, $\overline{pd}$는 예측값들의 평균이다.

$$SAD = \sum_i \left| (pd_i - \overline{pd}) \right| \qquad \text{식 (3)}$$

(그림 6)의 'Mixed'가 이러한 방법으로 생성된 결과로 다른 방법에 비해 압축률이 가장 높은 것을 볼 수 있다. 또한, 이 결과에서 실사 이미지라 하더라도 인덱스 방식의 8BPP 컬러 이미지에서는 일반적인 방법의 압축률이 좋으며, 그레이 이미지나 24BPP 등의 인접한 픽셀의 차이가 크지 않은 이미지에서 DPCM 을 적용하여 압축하는 것이 압축률을 높일 수 있음을 알 수 있다.



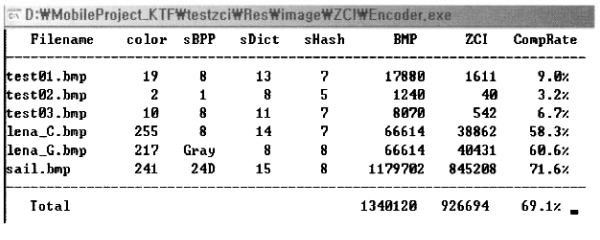
(그림 6) DPCM 예측 방법 비교

3. 모바일 게임 이미지 압축기의 구현

3.1 이미지 압축기와 복원 모듈 구현

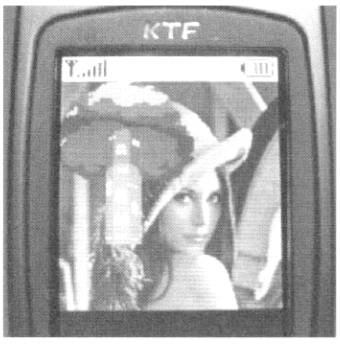
본 논문에서 제안한 모바일 이미지 압축의 개선 방법을 이용하여 실제로 이미지 압축기와 복원 모듈을 구현하였다. 이미지 압축기는 Windows XP 운영체제에서 Visual C/C++ 6.0 을 사용하여 구현하였고, 복원 모듈은 모바일 플랫폼인 BREW에서 C 언어로 구현하였으며, 에뮬레이터와 휴대전화 단말기 LG KP6160을 사용하여 실제 단말기에서의 동작 여부를 검증하였다.

(그림 7)은 본 논문에서 제안한 방법을 사용하여 이미지를 압축한 화면이고, (그림 8)은 압축된 Lena 이미지와 모바일 게임 이미지를 실제 휴대전화 단말기에서 복원하여 출력한 화면을 보이고 있다.


D:₩MobileProject_KTF₩testzci₩Res₩image₩ZCI₩Encoder.exe

Filename	color	sBPP	sDict	sHash	BMP	ZCI	CompRate
test01.bmp	19	8	13	7	17880	1611	9.0%
test02.bmp	2	1	8	5	1240	40	3.2%
test03.bmp	10	8	11	7	8070	542	6.7%
lena_C.bmp	255	8	14	7	66614	38862	58.3%
lena_G.bmp	217	Gray	8	8	66614	40431	60.6%
sail.bmp	241	24D	15	8	1179702	845208	71.6%
Total					1340120	926694	69.1%

(그림 7) 모바일 게임 이미지 압축기의 실행



(그림 8) 휴대전화 단말기에서의 동작

3.2 기존 압축 포맷과의 비교

본 논문에서 제안한 방법을 사용하여 압축한 이미지를 기존의 이미지 포맷인 GIF, PNG 와 BREW 플랫폼의 자체 압축 포맷인 BCI[8], 상용 모바일 이미지 솔루션인 SIS[9], SCI[10]와 비교하여 개선된 압축률을 비교하였으며, 그 결과를 <표 3>, <표 4>에 제시하였다. 이중 <표 4>에서는 SIS와 BCI에서 24BPP 이미지의 압축시 이미지의 손실이 발생하여 본 논문의 주제인 무손실 압축에서 벗어나므로 비교에서 제외하였으며, GIF와 SCI는 일반적으로 24BPP 이미지를 지원하지 않으므로 역시 비교에서 제외하였다.

<표 3> 제안된 방법과 기존 포맷과의 비교 (18×8~112×150)
(단위:Byte)

Image	GIF	PNG	BCI	SIS	SCI	Prop.
test1	2691	2576	2768	1751	1619	1611
test2	850	887	1140	123	59	40
test3	1686	1385	1646	635	548	542
test4	1312	1264	1496	488	410	409
test5	1563	1460	1744	739	635	634
test6	1236	1237	1455	442	372	285
test7	1115	1089	1340	348	263	262
합계	10453	9898	11589	4526	3906	3783
개선률	63.81%	61.78%	67.36%	16.42%	3.15%	-

<표 3>에서는 모바일 게임 이미지를 대상으로 제안된 방법을 기존 포맷과 비교하여 3.15%~67.36%의 개선 효과를 보였으며, <표 4>에서는 8BPP, 그레이 이미지, 24BPP 실사 이미지를 대상으로 기존 포맷과 비교하여 부분적인 개선 효과를 보였다.

<표 4> DPCM 적용시 실사 이미지에서의 비교(128×128~256×323)
(단위:Byte)

Image	BPP	GIF	PNG	BCI	SIS	SCI	Prop.
lena_G	Gray	19073	12113	16042	15025	14957	11441
peppers_G	Gray	18982	12140	16322	15304	15227	11560
baboon_G	Gray	20799	14659	16046	15028	14963	13927
sail	24BPP	-	107557	-	-	-	110825
serrano	24BPP	-	58248	-	-	-	57000

이 결과로 그레이 이미지나 24BPP 이미지 등의 인접한 픽셀의 차이가 크지 않은 이미지에서는 DPCM을 통한 예측 부호화 방법이 압축률 향상에 도움이 되는 것을 알 수 있다. 그러나, <표 4>를 볼 때 그레이 이미지에서는 이미지의 해상도가 작으므로 제안 방법의 압축률이 좀 더 높았으나 24BPP 이미지에서는 해상도가 그레이 이미지에 비해 높아지므로 기존 포맷인 PNG에 비해서는 압축률이 다소 떨어지는 것을 알 수 있었다. 그러나, 본 논문의 주요 주제는 8BPP 이미지의 압축률 향상에 관한 것이고, 또한 대부분의 모바일 게임에서는 픽셀 아트 기법을 사용하여 인접한 픽셀의 차이가 큰 8BPP 이미지를 사용하므로 24BPP 이미지 비교의 결과는 향후 연구 방향을 제시해주는 것으로만 결론을 내리고자 한다.

3.3 모바일 플랫폼에서의 복원 속도

구현된 모바일 이미지 복원 모듈을 실제 단말기에 올려서 복원 속도를 측정하였다. 본 논문의 주요 주제는 모바일 게임 이미지의 압축률 향상에 관한 것이지만 제안된 방법의 복원 속도가 모바일 플랫폼에서도 충분히 사용할 수 있음을 보이고자 하였다.

복원 속도 측정에는 모바일 게임 이미지와 컬러/그레이의 실사 이미지가 사용되었다. 복원 속도는 휴대전화 단말기와 에뮬레이터에서 각각 측정하였고, 결과는 (그림 9)에서 볼 수 있다. 에뮬레이터에서는 GIF를 지원하지 않아 결과에서 제외되었다.

결과를 검토해 보면 실제 단말기에서의 제안된 방법의 복원 속도는 약 900msec로 PNG의 약 2600msec에 비하면 빠르지만 GIF와 BCI의 속도인 약 300msec에 비하면 느린 것으로 나타났다. 또한 그레이 이미지의 경우 후처리 과정에서 DPCM 복원 과정이 필요하므로 컬러 이미지에 비해 속도가 저하되는 것을 볼 수 있다.

결과에서 제안된 방법이 GIF, BCI 등에 비해 속도가 저하되는 것으로 나타났는데, 이는 GIF, BCI, SIS는 휴대전화 단말기 자체에 내장된 모듈을 통해 동작하기 때문이다. 특히 BCI의 경우는 BREW 플랫폼에 내장된 압축 솔루션이므로 단말기의 운영체제 및 기본 API와 밀접하게 관련이 있어 다른 모듈에 비해 복원 속도가 빠르다.

이는 에뮬레이터에서 측정한 결과를 보면 좀 더 명확히 알 수 있는데, 에뮬레이터에서는 실제 단말기에서 측정한 결과와 다르게 제안된 방법과 BCI의 속도 차이가 없는 것으로 나타났다. 에뮬레이터인 만큼 약간의 오차와 각종 최적화 등의 요인이 있지만 그 보다 에뮬레이터에서는 BCI와 제안된 방법이 모두 외부 모듈의 형태로 구동되는 것이 단말기와 에뮬레이터 속도 차이의 중요한 이유이다.

PNG의 경우는 포맷 자체의 복잡함으로 인해 모든 환경에서 제안된 방법보다 속도가 느린 것으로 분석되었으며, 제안된 방법의 복원 모듈이 같은 외부 모듈을 사용하는 SCI에 비해서 상대적으로 속도가 느린 점은 향후 연구 과제로 남아있다.

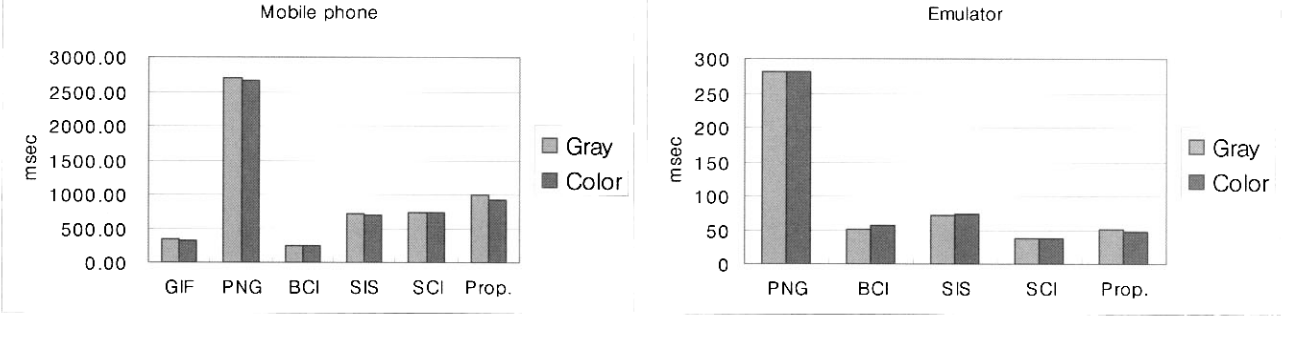
4. 결 론

본 논문에서는 모바일 게임의 전체 용량에서 상당 부분을 차지하는 이미지를 무손실로 압축함에 있어 실제 압축을 수행하기에 앞서 전처리 과정을 거쳐 압축률을 향상시키는 방법을 제안하였다. 전처리 과정에서는 사전 기반 부호화의 특징인 사전의 크기를 이미지의 BPP와 비트맵의 크기 등에 맞도록 가변적으로 설정하고 픽셀 패킹 및 DPCM 기법을 사용하여 보다 압축률을 높일 수 있었다.

제안된 압축 방법을 기존 포맷과 비교하여 상용 모바일 이미지 솔루션에 비해서는 약 9.7%, PNG 등의 기존 포맷에 비해서는 약 64.3%의 압축률이 향상됨을 보였으며, 실사 이미지에서도 8BPP 이미지의 경우에서는 다소 효과가 있음을 보였다. 또한, 모바일 플랫폼에서 복원 속도를 측정하여 실제 휴대전화 단말기에서도 무리 없이 사용할 수 있음을 보였다.

제안된 방법은 특정 플랫폼에 종속되지 않고 일반적인 방법을 사용하여 높은 이식성으로 다양한 플랫폼에서의 적용 가능성도 보였으며 내부적으로 이미지의 비트맵 데이터를 쉽게 접근할 수 있어 게임 이미지에 회전이나 알파 블랜딩 등의 특수 효과 처리도 쉽게 가능하다.

향후에는 복원시 전체적인 속도 향상을 위한 개선 방안과 24BPP 이미지 등에서 압축률을 높일 수 있는 방안, 좀 더 다양한 전처리 단계에 대한 연구, 늘어나는 3D 게임 등에 사용할 수 있는 손실 압축에 관한 부분이 연구되어야 할 것이다.



(그림 9) 모바일 플랫폼에서의 복원 속도 비교

참 고 문 헌

[1] "모바일 게임 이용자 현황", (재)한국게임산업개발원, pp.25-26, 2005.

[2] S.W. Golomb, "Run-length encoding", IEEE Trans. Inform. Theory, vol.IT-12, pp.399-401, 1966.

[3] J.M. Gilbert, R.W. Brodersen, "A Lossless 2-D Image Compression Technique for Synthetic Discrete-Tone Images", Proc. Data Compression Conf., pp.359-368, March 1998.

[4] QUALCOMM, http://brew.qualcomm.com

[5] Sun Microsystems, "MIDP(JSR-37) Specification 1.0", September, 2000.

[6] 최재영, "휴대전화 단말기를 위한 개선된 이미지 처리 알고리즘의 구현", 한국해양대학교 대학원 석사학위논문, 2003.

[7] 황병연, 오명석, "모바일 게임을 위한 압축 기술", 정보처리학회지, 제9권, 제3호, pp.49-54, 2002년 5월.

[8] QUALCOMM, "BREWTM 2.1 Compressed Image Authoring Guide", May 5, 2003.

[9] NeoMtel, SIS(NeoMtel's mobile multimedia solution), http://www.neomtel.com

[10] Softzen, SCI(Softzen's mobile compression solution), http://www.softzen.co.kr

[11] J. Ziv, A. Lempel, "A universal algorithm for sequential data compression", IEEE Trans. Information Theory, Vol.IT-23, pp.337-343, 1977.

[12] 진용선, "실시간 데이터 압축을 위한 Lempel-Ziv 알고리즘의 하드웨어 설계", 한양대학교 대학원 박사학위논문, pp.11-12, 2001.

[13] P. Deutsch, "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, Aladdin Enterprises, May, 1996.

[14] ITU-T Recommendation H.262 (1995), H.263 (1996).

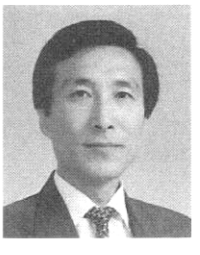
[15] 이성수, 채수익, "저해상도 양자화된 이미지를 이용하여 연산량을 줄인 움직임 추정기법", 전자공학회논문지, Vol.33, No.8, pp.81-88, 1996.

김 세 웅

e-mail : zenina@netian.com

2003년~현재 서울산업대학교 컴퓨터공학과 석사과정

관심분야 : 게임, 화상처리, 운영체제, 시스템 소프트웨어



조 병 호

e-mail : jobh@snut.ac.kr

1978년 고려대학교 공과대학 전자공학과 (공학사)

1983년 서울대학교대학원 컴퓨터공학과 (공학석사)

1992년 서울대학교대학원 컴퓨터공학과 (공학박사)

1977년~1981년 금성통신㈜ 연구소 소프트웨어연구실 연구원

1984년~현재 서울산업대학교 컴퓨터공학과 교수

관심분야 : 운영체제, 분산처리, 시스템 소프트웨어