데이타의 유사성을 이용한 실제 색상 화상의 압축 기법

김 태 윤† 송 길 영†† 조 광 문††† 최 창 원†††

요 약

컬러 화상의 실제 색상은 한 픽셀을 나타내는 3가지 기본 색상 Red, Green, Blue에 각각 1 바이트씩 할당된 것을 말한다. 실제 색상을 사용하면 거의 모든 색상을 표현할 수 있는 장점을 갖고 있다. 그러나 많은 기억 용량이 요구되어 효율적인 화상 압축 기법이 필요하게 된다. 기존의 화상 압축 기법들은 대부분 실제 색상 화상의 특성을 고려하지 않은 방법을 사용하여 효율적인 압축을 수행할 수 없었다. 실제 색상 화상의 각 픽셀과 인접 픽셀들과의 차이는 미소하다. 각 픽셀을 나타내는 3바이트의 데이타에서 몇개의 비트만이 다른 값을 갖게 된다. 그러나 서로 다른 비트들이 3바이트 내에 흩어져 있어서 효율적인 압축 결과를 얻을 수 없다. 따라서 본 연구에서는 실제 색상 화상의 구조를 변경시켜 데이타의 유사성을 증가시킨다. 이를 이용하여 효율적인 압축을 수행할 수 있는 기법을 제안하였다.

A Compression Method for the True Color Images Using the Similarities of Data

Tai Yun Kim,† Gil Young Song,†† Kwang Moon Cho††† and Chang Won Choi†††

ABSTRACT

It is the true color that is allocated 1 byte to three basic colors, red, green, and blue in order to represent a pixel of an image. It is a merit that almost all colors are representable by using the true color. However, it requires a lot of storages. An efficient image compression method is necessary. Most of the existing compression methods have not considered the characteristics of the true color images. Therefore, the efficient compression has been almost impossible. The differences of each pixel and its adjacent pixels in the true color images are few. Only a few bits in 3 bytes data which represent a pixel have different values. But the different bits are scattered in 3 bytes, so efficient compression results are not achieved. Therefore, in this study it is shown that the similarities of data are increased by relocating the data structures of true color images. And an efficient compression strategy which uses the similarities of data is proposed.

1. 서 론

컴퓨터 그래픽에서 화상을 표현하는 방법은 크게 벡터 방식과 래스터 방식으로 구분한다. 벡터 방식은 화상의 구성 요소를 단위로 정보를 저장하는 방식으로서 이를 지원하는 디스플레이를 캘리그래픽 디스플레이(calligraphic display)라고 한다. 래스터 방식은 화상의 구성 요소와는 무관하게 하나의 장면 단위로 정보를 저장하는 방식이며, 그 정보는 점들의 배열로 이루어진다. 벡터 시스템은 래스터 시스템에 비하여 화상의 수정이나 각종 작업시에 정보를 비효율적으로 다루므로 일반적으로 래스터 시스템이 더 많이 사용된다[1]. 래스터 시스템은 전체 화상의 표현시에 화상의 구조와는 무관한 비트맵 단위를 사용하므로 비트맵 그래픽스(bitmapped graphics)라 한

† 종신회원 : 고려대학교 전산과학과 교수
†† 정 회 원 : 한국무역정보통신 연구원
††† 정 회 원 : 고려대학교 전산과학과 박사과정
논문접수 : 1994년 5월28일, 심사완료:1994년 9 월14일

다[2, 3].

컴퓨터 그래픽스에서는 기본 색상을 더하는 작업을 통하여 모니터에 색상을 표현한다. 컴퓨터에서 더해지는 색상은 각 색상에 대하여 세가지의 값, Red, Green, Blue로 표현된다. 따라서 이를 RGB 색상이라고 부른다.

비트맵 그래픽스에서 색상의 수는 표현시에 요구되는 비트의 수에 비례한다. 기초 색상인 RGB 각각에 1 바이트씩을 할당하는 방식을 실제 색상(true color)이라 하며 동시에 1670여만 가지(2^{24})의 색상을 표현할 수 있다. 한 픽셀당 요구되는 정보가 24 비트이므로, 1024*1024 크기의 실제 색상 화상을 저장하려면 25,165,824 (1024*1024*24) 비트가 필요하게 된다. 실제 색상은 많은 수의 색상을 동시에 표현할 수 있으나 저장시에 많은 기억 장소를 차지하며 정보 전송시에 회선의 점유 시간이 늘어나는 문제점이 발생한다. 따라서 실제 색상 화상의 압축이 요구된다.

지금까지는 컴퓨터 그래픽을 다루는 소프트웨어 업체들이 고유의 화상 표현 형식을 정의하고 그에 맞는 압축 방식들을 제공하여 사용하였다. 이 방식들은 대부분 RLE(Run Length Encoding) 방식에 기반을 둔 것이었다. RLE 방식은 화일의 특수한 형태, 즉 연속되는 같은 심볼의 나열이 많을 때 효과적으로 적용될 수 있는 방법이다. 그러나 실제 색상 화상의 경우에는 반복되는 데이타의 양이 급격히 감소하므로 좋은 압축률을 거둘 수 없다. 또한 화상이 컴퓨터를 이용하여 그려진 경우보다 스캐너와 같은 장비를 이용하여 입력된 경우는 반복의 횟수가 더욱 큰 폭으로 감소하게 된다.

본 연구는 24 비트 실제 색상 화상의 비손실 압축에 관한 것이다. 실제 색상은 매우 많은 수의 색상을 동시에 사용할 수 있으므로 미소하게 차이나는 색상을 모두 표현할 수 있다. 인접한 각 픽셀들이 미소한 차이를 갖는다는 특성, 즉 각 픽셀간의 데이타의 유사성을 이용한다. 각 픽셀을 구성하는 정보의 내용을 재배치 함으로써 압축률을 높일 수 있는 기법을 제안하였다.

2. 화상 압축

실제 색상의 데이타 특성을 분석하기 위하여 지금까지 컴퓨터 그래픽에서 사용되어 온 여러가지의 화상 표현 방식을 분석하고, 화상의 압축 기법을 제시한다.

2.1 화상 표현 형식

화상의 표현 형식 중에서 완벽히 24 비트의 실제 색상을 지원하는 것으로는 BMP 형식과 비손실 압축을 수행하는 24 비트 PCX, TGA, 그리고 TIFF 형식 등이 있다. 손실 압축을 수행하는 형식에는 DCT(Discrete Cosine Transformation)를 기반으로 하는 JPEG이 표준으로 사용되고 있다.

(1) BMP 형식

윈도우즈(MS-Windows) 3.0의 페인트 브러쉬(paint brush)에서 지원하는 형식이다. 이 형식은 최대 24 비트 형식까지 수용하며 압축되지 않은 형태로 저장되는 화일 형식이다.

압축을 수행하지 않는 이유는 두가지다. 첫째, 화상이 압축되어 있지 않으므로 복원 과정이 필요없어 화일을 빨리 읽어들일 수 있다는 것이다. 둘째, 보통 스캐닝(scanning)된 화상을 다루는 경우가 많기 때문이다. 이 경우에 압축된 결과가 압축 전의 크기보다 커지는 네거티브(negative) 압축이 빈번하게 발생한다. 사용자의 입장에서는 이용하기 쉬운 화일 형태이지만, 데이타의 크기가 커지는 단점이 있다.

(2) 24 비트 PCX

PCX 화상 화일 형식은 ZSoft 사에서 그래픽 데이타의 화일화를 위해 만든 것이다. 현재 대부분의 소프트웨어가 지원하고 있는 형식으로서 광범위하게 사용되고 있다. PCX 화상 화일 형식은 컬러 및 흑백을 모두 제공한다. PCX 화일 형식은 헤더가 128 바이트로 이루어진 고정 헤더로 구성된다. 그리고 그 이하는 화상 데이타로 구성

된다. 특히 주의할 점은 화상 데이타는 반드시 바이트 단위에 의한 RLE 방식으로 압축된 것이어야 한다는 것이다. PCX 헤더에는 그림의 좌표와 크기, 팔레트, 사용된 플레인 수 등 화상의 특징에 대한 정보가 들어 있다.

24 비트 PCX는 PCX의 확장 형태다. 이는 24 비트의 이미지를 다루며 비트 평면 단위로 RLE 방식의 압축을 수행한다.

(3) TGA 형식과 TIFF 형식

Truevision의 Targa 보드가 지원하는 형식으로 1, 8, 16, 24, 32 비트의 색상을 지원한다. 이 형식은 행 단위로 간단한 RLE 방식의 압축을 수행하거나 또는 압축을 수행하지 않는다. TGA 형식은 데이타를 저장할 때 일반적인 화상 방식의 역순인 Blue, Green, Red의 순으로 저장한다.

TIFF 형식은 화상의 모든 가능한 특징들을 포함할 수 있도록 정의되어 유연하고 확장이 가능한 형식이다. TIFF(Tagged Image File Format)의 원래 설계 방향은 탁상 출판 패키지와 같은 문서나 출판에 관계되는 응용 프로그램과 화상 스캐너나 그래픽 응용 프로그램의 화상 정보 입력 및 통합 환경을 제공하는 것이었다.

(4) JPEG 형식

손실 압축에서 화상의 품질은 주관적인 판단에 따라 달라지는데 이는 압축률의 영향을 받는다. 또한 각 화상의 특성에 따라서도 압축률은 달라진다. 디지탈 화상의 응용을 위해서는 화상 압축 기법뿐 아니라 이를 위한 국제적인 표준이 제정되어 이를 만족시키는 화상 처리 장치들이 제작되어야 한다. 그래서 화상 처리 장치 사이의 호환성을 유지할 수 있어야 한다.

이에 대한 표준화 작업이 JPEG(Joint Photographic Experts Group)에 의하여 진행되어서 표준화안의 이름이 통상적으로 JPEG이라고 불리운다. JPEG은 그레이 화상 및 컬러 화상의 디지탈 압축에 관한 국제 표준이다.

2.2 화상 압축 방식

화상 압축 방식은 손실(lossy) 압축과 비손실(lossless) 압축으로 분류할 수 있다. 손실 압축은 압축시에 화상의 중요 부분을 부각시키고 다른 필요치 않은 부분의 손실을 감수하여 압축한다. 따라서 손실 압축은 기준이 되는 손실의 비율에 따라서 압축률과 해상도가 영향을 받게 된다[4]. 비손실 압축은 텍스트 화일의 경우와 같이 이진 부호화된 화상의 데이타를 전혀 잃어버리지 않고 다시 원래의 상태로 복원시킬 수 있는 방식이다.

손실 압축은 압축률이 높지만 데이타를 잃어버리는 단점이 있고, 비손실 압축의 경우는 압축률은 비교적 떨어지지만 데이타를 잃는 경우는 없다. 따라서 먼저 손실 압축을 통하여 압축된 데이타를 다시 비손실 압축을 통하여 압축시키는 방법이 사용되기도 한다.

(1) 손실 압축 방식

예측 부호화 기법, 변환 부호화 기법이 주로 사용되고 있으며 이들의 변형이나 조합으로 성능 향상을 시도한다. 손실 압축의 적용 분야는 크게 정지 화상용 압축 알고리즘과 동화상용 압축 알고리즘으로 나눌 수 있다. 정지 화상은 또한 이진 정지 화상과 컬러 정지 화상으로 나뉘어진다.

이진 정지 화상의 압축을 위한 표준안으로는 CCITT의 추천안 T.4 및 T.6의 MH(Modified Huffman) 부호화 방식, MR(Modified Read) 부호화 방식, MMR(Modified Modified Read) 부호화 방식 등이 널리 사용된다. 또한 이진 정지 화상에 대한 국제 표준안을 제정하기 위하여 1988년 JBIG(Joint Bi-Level Image Experts Group)이 형성되었다. 이진 정지 화상 압축 알고리즘은 주로 팩시밀리 전송에 사용된다.

그레이 화상 및 컬러 정지 화상 압축 기법의 표준은 ISO와 CCITT에 의해 1986년에 설립된 JPEG에 의하여 제정되었다. 그레이 화상 및 컬러 정지 화상 압축 알고리즘은 영상 데이타베이스, 비디오텍스, 컬러 팩시밀리 등에 사용된다[5].

동화상 압축 기법은 1988년에 설립된 MPEG (Moving Picture Experts Group)에 의하여 표

준화 작업이 이루어졌다. 동화상 압축 기법은 응용 분야를 기준으로 저해상도와 고해상도로 구분된다. 저해상도는 비디오 전화나 원격 회의 시스템 등에 사용되며 고해상도는 고화질 텔리비젼용 등으로 사용된다.

(2) 비손실 압축 방식

통계적 모델을 이용한 압축 기법인 세논-페노(Shannon-Fano) 부호화 방식과 허프만(Huffman) 부호화 방식, 그리고 허프만 부호화 방식에서 압축률을 반감시키는 요소를 해결한 동적 허프만 코드화 방식 등이 있다[6, 7]. 또한 LZ(Limpel-Ziv) 기반의 압축이나[8, 9], 연속되는 같은 심볼의 나열이 많이 발생할 때 사용하면 효과적인 방법인 RLE 방식, 그리고 유사한 데이타의 경우 반복되는 부분을 생략하는 압축 방식인 differential 압축 방식 등도 사용된다[10].

3. 알고리즘의 설계 및 구현

본 연구에서 제안한 압축 알고리즘은 화상 데이타의 유사성을 이용하기 위하여 먼저 데이타를 재배치하여 각 픽셀들 간의 유사성을 향상시키고 압축을 수행한다. 제안한 압축 알고리즘의 효율성을 위해 두가지의 압축 기법을 적용한다. 데이타의 일치성을 이용하는 RLE 기법과 데이타의 유사성을 이용하는 differential 압축 기법을 화상 데이타의 블럭 단위로 적용한다.

3.1 데이타의 재배치

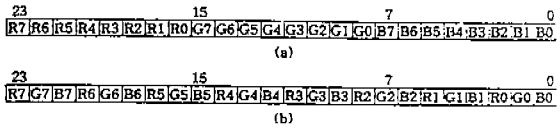
실제 색상으로 표현된 화상을 분석해 보면 서로 인접한 픽셀들은 유사한 색상을 갖고 있음을 알 수 있다. 일반적으로 색상의 차이가 많이 나타나는 부분은 서로 다른 대상의 경계 부분에 해당하고, 동일한 대상을 표현하는 부분은 대개가 유사한 색상으로 이루어진다. 예를 들어, 하늘이나 숲을 나타내는 화상에서 각 픽셀과 인접한 픽셀과의 색상 차이가 미세하여 거의 유사한 색상을 갖게 된다. 본 연구에서는 이러한 데이타의 지역적인 유사성을 이용한 효율적인 압축 기법을 제안하였다.

기존의 화상 압축 방식들은 대부분 단순한 RLE 방식이거나 LZ 압축 기법들이 사용되었다. 이는 데이타의 일치성에 그 관점을 둔 것으로서 화상을 표현하는 색상의 수가 최대 16이나 256인 경우에는 적절하게 사용될 수 있다. 이러한 방식에서는 표현이 불가능한 색상은 가장 유사한 색상으로 표현하였기 때문에 인접한 픽셀 간의 색상 차이가 없게 된다. 그러나 실제 색상 화상의 경우에는 이러한 기법들이 효율적이지 못하게 된다. 실제 색상은 인간이 거의 구별해 내지 못할 정도의 많은 색상(1670 여만가지)을 사용하므로 동일한 색상의 픽셀이 많으리라고 기대할 수 없다. 더우기 스캐너와 같은 장비를 이용하여 입력된 화상의 경우에는 거의 일치하는 픽셀이 발생하지 않는다. 대신에 인접한 픽셀들 간의 색상 차이가 미소하다는 특성이 있다. 따라서 이러한 특성인 데이타의 유사성을 잘 이용할 수 있는 기법을 개발한다.

기존에 사용되던 differential 압축 기법은 유사한 문자열과 같은 대상에 대하여 사용하던 것이다. 실제 색상의 경우에는 하나의 픽셀을 표현하는 데이타의 양이 3바이트다. 인접한 픽셀이 유사한 색상을 갖게 되므로, 그 픽셀의 색상 정보를 나타내는 3바이트의 데이타도 유사한 값을 갖게 된다. 따라서 그 3바이트를 기준으로 differential 압축 기법을 사용하여도 충분한 효과를 얻을 수 있으리라 예상할 수 있다.

데이타의 유사성을 이용하기 위해서는 데이타가 유사한 형태를 갖도록 변형할 필요가 있다. 실제 색상 데이타가 저장되는 방식은 각 픽셀의 RGB 요소 각각에 1바이트를 할당하여 저장한다. 따라서 세가지 기초 색상의 인덱스 중 하나만 바뀌어도 데이타의 유사성이 매우 감소하게 된다. 본 연구에서는 이러한 세가지 기초 색상을 표현하는 데이타의 구조를 변형하여 인접한 픽셀 간의 유사성을 증가시킨다. RGB 각각을 표현하는 인덱스들을 비트 단위로 배열하여 하위 비트들끼리 하나의 묶음을 이룰 수 있도록 한다. 이

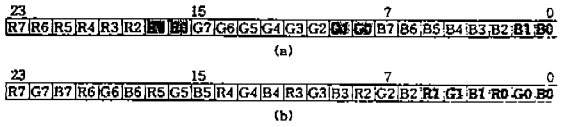
러한 데이타의 재배치 전후의 각 픽셀의 비트 배열을 (그림 1)과 같이 나타낼 수 있다.



(그림 1) 재배치 전후의 비트 배열
(Fig. 1) Bits arrangements of before & after relocation

각 픽셀에 대한 정보를 나타내기 위해 현재 (그림 1(a))와 같이 RGB 각각에 1바이트 씩을 할당하던 것을, (그림 1(b))와 같이 RGB 각 비트들을 교차하여 배열하는 것을 의미한다.

이러한 재배치의 잇점을 (그림 2)와 같이 나타낼 수 있다. 인접한 픽셀의 RGB 값들이 각각 마지막 2비트씩 변화하였다고 할 때, (그림 2 (a))는 각 바이트의 상위 6비트만이 앞의 데이타와 일치한다. 그러나 본 연구에서 제안한 데이타의 재배치를 나타낸 (그림 2(b))에서는 3개의 바이트 중에서 하위 6비트만이 변화하게 되어 3바이트 단위로 데이타를 처리하게 될 경우에 많은 양의 데이타가 중복되어 압축 과정에서의 잇점을 얻을 수 있다.



(그림 2) 재배치 전후의 비트 변화
(Fig. 2) Bits changes of before & after relocation

이러한 재배치의 결과 각 픽셀 간의 데이타의 유사성이 증가된다. 각 픽셀이 실제로 같은 색상을 갖는 경우에는 3바이트 모두 일치하는 경우로서 기존의 기법과 다른 점이 없다. 그러나 1바이트도 일치하지 않는 경우가 줄어들고, 1바이트 또는 2바이트가 일치하는 경우가 증가된다. 즉, 변환 후의 데이타들이 변환 전의 데이타에 비하여 많은 중복성을 갖게 된다는 것이다. 따라서 이러한 점을 압축 알고리즘에서 이용할 수 있다.

재배치 작업에 따른 시간 지연이라는 오버헤드가 발생한다. 이를 해결하기 위한 2가지 방법이 있을 수 있다. 첫째, 재배치 과정 자체를 하드웨어로 구현하는 방법이다. 둘째, 이러한 방식을 하나의 화상 표현 형식으로 제정하는 것이다.

3.2 압축 알고리즘

Differential 압축 방법은 압축을 하는 과정에서 이전에 나왔던 부분이 중복되는 경우에 중복되는 부분을 제외한 비중복 부분만을 기술하는 방식이다. 본 연구에서는 이 differential 압축 방법을 사용한다.

본 연구에서 제안한 압축 알고리즘은 먼저 한 픽셀의 색상 정보를 나타내는 3바이트를 읽고, 다음 픽셀의 3바이트를 읽어서 먼저 읽어들인 것과의 차이를 알아낸다. 이를 바이트 단위로 수행할 수도 있지만, 여기에서는 3비트 단위로 수행한다. 비트 단위의 연산이 되면 수행 시간이 길어지는 단점이 있지만, 압축률을 보다 향상시킬 수 있는 장점이 있다. 3비트 단위로 데이타의 일치성을 구분한다. 이렇게 되면 모두 9가지의 경우가 나와 각 경우를 나타내기 위한 4비트의 태그가 필요하게 된다. 그러나 한가지 경우를 제외하면 태그 비트로서 3비트만 사용하여도 된다. 따라서 3비트의 태그를 작성한다. 태그 비트의 내용은 〈표 1〉과 같다.

〈표 1〉 압축을 위한 태그 비트
〈Table 1〉 Tag bits for compression

태그	의 미
000	3 개의 바이트 내용이 모두 다르다.
001	상위 6 비트의 내용이 같다.
010	상위 9 비트의 내용이 같다.
011	상위 12 비트의 내용이 같다.
100	상위 15 비트의 내용이 같다.
101	상위 18 비트의 내용이 같다.
110	상위 21 비트의 내용이 같다.
111	3 개의 바이트 내용이 모두 같다.

이 방법을 사용하면 부분적으로 반복되는 데이타에 대한 효율적인 압축을 기대할 수 있지만 3바이트가 완전히 반복되어도 3비트의 태그가 필요하다. 따라서 완전히 반복되는 경우를 효율적

으로 압축시키는 방법으로 RLE 기법을 추가적으로 적용한다.

화상에서 동일한 색상이 반복되는 경우는 보통 일부분의 구간에서 발생한다. 이처럼 동일한 색상이 반복될 경우에는 differential 압축보다 RLE 방식을 적용하는 것이 더 효율적이다. 이 방식은 [Pcomp+C]로 데이타를 표현하여 많은 수로 반복하는 데이타를 한번에 표현한다. 여기에서 Pcomp는 압축하고자 하는 부분이고 C는 반복 횟수를 나타낸다.

이처럼 하나의 화일에 두가지의 기법을 적용하기 위해 고려할 사항이 있다. 먼저 데이타의 특성을 분석하여 어느 부분에 어떤 기법의 알고리즘을 적용할 것인지를 결정하는 전처리 과정이 필요하다. 그리고 선택된 부분이 어떤 알고리즘을 이용하여 압축되었나 하는 정보를 저장하여 압축된 화일의 복원시에 사용할 수 있어야 한다.

어떤 부분에 어떤 알고리즘을 사용할 지는 압축을 수행하기 전에 압축하고자 하는 목적 화일의 내용을 분석하여 결정한다. 어떤 알고리즘이 특정 부분에 사용되었는가는 화일의 머리 부분에 그에 대한 정보를 헤더로 저장하여 복원할 때 사용한다.

이 과정은 먼저 압축하고자 하는 화일을 적당한 크기의 블럭들로 나누고 분석하여 어떤 알고리즘을 사용하는 것이 적합한지를 판별한다. 데이타가 모두 동일하게 반복되는 경우가 매우 많은 경우에는 RLE 방식을 적용하는 것이 좋다. 반대로 동일하게 반복되지는 않으나 유사한 데이타가 많은 경우에는 differential 압축 방식이 더 적합할 것이다. 또한 급격히 변하는 데이타의 양이 많은 경우에도 RLE 방식이 적합하다. Differential 압축은 모두 같지 않은 데이타의 경우에 네거티브 압축이 발생할 수 있다. RLE 방식은 그러한 경우에 압축이 되지 않은 형태로 기록하여 네거티브 압축을 방지할 수 있기 때문이다.

실험을 통한 임계값, 즉 동일한 데이타의 수와 완전히 다른 데이타의 수를 기준으로 각 블럭에 대한 적용 알고리즘을 결정한다. 그 내용을 압축을 수행하기 전에 압축되는 화일의 헤더 부분에 저장한다.

RLE 방식의 압축은 특수 문자(SC:Special Character)를 사용하는 경우와 그렇지 않은 경우가 있다. SC를 사용하지 않는 경우는 [D+C]로 표기한다. 여기에서 D는 3바이트의 데이타, C는 반복 횟수를 나타낸다. 이것은 반복하지 않는 데이타도 먼저 반복 횟수에 0을 저장하여 1바이트를 낭비하게 되는 단점이 있다. SC를 사용하는 경우는 화일내에서 나타나는 빈도가 적은 데이타를 SC로 정한다. 반복되는 내용이 존재하는 경우에는 [SC+D_{sc}+C]로 표기한다. D_{sc}는 SC로 시작되는 3바이트의 데이타를 나타낸다. 반복되지 않는 경우에는 [D] 만을 표기한다. 반복되지 않는 데이타가 SC로 시작하는 경우에는 문제가 발생하므로 [SC+D_{sc}+00000000]으로 표기한다. 따라서 최대 256회의 반복까지 처리할 수 있다. RLE 방식의 데이타 형태는 〈표 2〉의 4가지 경우 중의 하나가 된다.

〈표 2〉 RLE 방식의 데이타 종류
〈Table 2〉 Data categories of RLE method

	데이타의 반복 여부	SC로 시작하는지 여부
A	×	×
B	×	○
C	○	×
D	○	○

A : [D]
B : [SC + Dsc + 00000000]
C : [D + C]
D : [SC + Dsc + C]

3.3 압축 기법의 선택 방법

본 연구에서 제안한 압축 알고리즘에서는 두가지의 기법을 선택적으로 적용하였다. 이때 알고리즘 선택의 기준은 하나의 블럭내에서 앞의 픽셀과 일치하는 픽셀의 갯수와 불일치하는 픽셀의 갯수다. 블럭의 크기는 실험을 통한 기준으로 선택된다. 일단 블럭의 크기가 정해지면 그에 따라서 알고리즘 선택의 임계값으로 사용될 수치들, 즉 일치 픽셀의 갯수, 불일치 픽셀의 갯수 등이

정해지게 되는데 이것 역시 실험을 통하여 얻은 결과 수치들이다.

압축 알고리즘을 선택하기 위하여 압축하고자 하는 화일의 특성을 먼저 분석한다. 따라서 실제 압축을 수행하기 전에 데이타 화일을 한 번 읽어 들이는 1패스의 과정이 요구되므로 전체 과정은 2패스가 된다.

하나의 블럭을 이루는 데이타가 비교적 큰 단위로 구성된다면 전체 블럭을 전부 읽지 않고 일부분으로 블럭의 특성을 결정지어 적용할 기법을 선택할 수 있다. 제안한 알고리즘에서 RLE 방식을 적용하는 경우는 일치하는 화소의 수가 매우 많은 경우와 유사성이 거의 없는 데이타가 대부분을 차지하는 경우다. 만약 한 블럭의 데이타를 읽어 나가다가 일치하는 픽셀의 수가 블럭당 요구되는 일치 픽셀의 수보다 작으리라는 것을 예상할 수 있다면 더 이상 그 블럭을 읽을 필요가 없게 된다. 동시에 유사성 있는 픽셀들의 수도 계속 세어 나가야 한다.

이를 식으로 표현하면 다음과 같다.

$N_T - N_{NS} < N_{RS}$ ····· [식 1]

$N_T - N_S < N_{RNS}$ ····· [식 2]

N_T : 전체 픽셀의 수

N_{NS} : 이미 읽은 불일치 픽셀의 수

N_S : 이미 읽은 유사성이 있는 픽셀의 수

N_{RS} : RLE방식을 적용하기 위해 블럭당 요구되는 일치 픽셀의 수

N_{RNS} : RLE방식을 적용하기 위해 블럭당 요구되는 유사성이 거의 없는 픽셀의 수

[식 1] 또는 [식 2]를 만족하는 블럭은 더 이상 읽을 필요없이 differential 방식을 선택하고 다음 블럭을 처리하면 된다. 이러한 2패스의 알고리즘을 1패스의 알고리즘으로 바꿀 수는 없으나, 첫번째 패스의 처리로 인하여 수행 시간의 상당한 절약을 얻을 수 있다.

4. 평가 및 분석

압축 알고리즘에 대한 성능 평가에서 고려할 사항으로는 압축률 외에도 압축 알고리즘의 복잡성, 복원의 용이성, 에러 발생률 등이 있다. 이 중에서 압축 알고리즘의 수행 단계 분석과 압축률에 대한 분석을 제시한다.

4.1 압축 알고리즘의 수행 단계 분석

RLE 방식은 한번의 패스로 압축이 가능한 간단한 방법이다. 화일을 읽어나가면서 반복되지 않는 데이타가 나오면 그 이전까지 반복된 횟수를 기술하므로 가장 빠른 방법이다. RLE 방식은 압축의 복원시에도 반복 횟수와 심볼의 한 단위를 읽는다. 그리고 반복 횟수만큼 심볼을 화일에 쓰는 것으로 수행이 끝나게 된다.

LZ 압축은 심볼을 입력받으면서 압축을 수행하는 1패스 알고리즘이다. 압축을 수행해 가면서 사전을 구축한다. 사전을 구축하는 방법은 심볼을 읽어서 사전에 존재하면 그 위치를 가리키는 포인터를 출력 화일에 넣고, 만약 사전에 없으면 사전에 새로운 심볼로 입력한다. 따라서 LZ 알고리즘의 주요 수행 시간은 사전의 탐색 시간과 사전과 출력 화일을 동시에 갱신하는 과정이다. 이러한 LZ 압축 방식은 데이타가 일정한 범위 내에서 중복성을 가져야 좋은 효율을 얻을 수 있다.

본 연구에서 제안한 압축 알고리즘은 2번의 패스를 필요로 한다. 제안한 알고리즘에서는 differential 알고리즘과 RLE 방식을 선택적으로 적용하기 위해서 먼저 전체 화일을 읽어서 어떤 알고리즘을 적용할 것인가를 결정한다. 그리고 블럭 단위로 적용 알고리즘을 수행한다. 2가지 압축 방식을 한 알고리즘에 적용하기 때문에 다른 알고리즘과는 달리 2번의 패스가 필요하다.

제안한 알고리즘의 주요 수행 시간은 전체 화일을 2번 읽는 과정과 differential 알고리즘에서의 반복적인 비교 연산 과정이다. Differential 알고리즘에서는 바로 전에 입력받은 데이타와 지금 입력받은 데이타 사이의 차이를 구하여 일치하지 않는 부분을 화일에 기록한다. 따라서 RLE 알고리즘과 비교하면 하나의 심볼에 대한 비교

횟수가 늘어나게 된다. 제안한 방식에서는 일정한 횟수의 비교가 정량적으로 이루어지는 것이 아니라 비교치를 기준점으로 만족하는 경우 비교를 중단한다. 따라서 일률적으로 말할 수는 없으나 RLE 방식에 비하여 평균 3배 정도의 비교 횟수가 증가한다. RLE 방식을 적용하는 블럭에서는 3 바이트의 심볼을 한 단위로 처리하므로 한 바이트를 기준으로 처리하는 텍스트 기반의 RLE 방식과 비교하면 비교의 횟수가 1/3로 줄어들게 된다. 결과적으로 제안한 알고리즘의 수행 시간은 LZ 방식보다는 빠르고 RLE 방식보다는 느리게 된다.

4.2 압축률의 비교

본 연구에서 제안한 방법과 기존의 방법을 사용하여 24비트 실제 색상 화상 데이타에 대하여 압축을 수행하였다. 제안한 알고리즘을 재배치 전후의 실제 색상 화상에 적용하고, 기존의 화상 표현 형식들로 표현된 화일과 압축률을 비교, 평가하였다.

압축률은 다음의 식을 사용하여 계산하였다.

$$\text{압축률} = \frac{\text{압축 후의 화일 크기}}{\text{압축 전의 화일 크기}} * 100$$

〈표 3〉은 동일한 화상에 단계별로 화상 처리를 한 후에 압축을 수행한 결과이고, (그림 3)은 이를 화일의 크기를 기준으로 그래프로 나타낸 것이다.

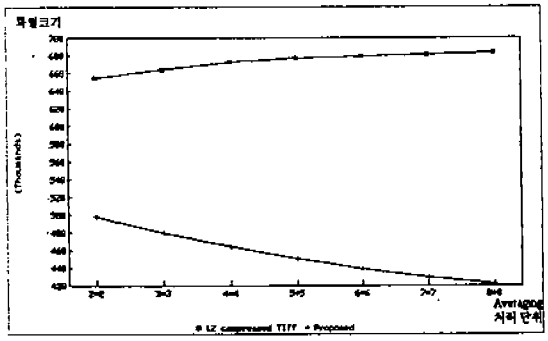
〈표 3〉은 윈도우즈 환경에서 Photo Styler를 이용하여 화상에 2*2, 3*3, 4*4, 5*5, 6*6, 7*7, 8*8 방식으로 averaging 처리를 수행한 것이다. Averaging 처리의 결과 화상에서 인접한 화소간의 데이타가 모두 일치하는 경우가 점차 줄어든다. 그러나 데이타의 일부분이 일치하는 수가 매우 많이 늘어나고, 전혀 일치하지 않는 데이타의 수가 큰 폭으로 줄어든다. 따라서 제안한 방식의 알고리즘을 적용할 때 충분한 잇점을 제공한다.

결과적으로 〈표 3〉에서 볼 수 있듯이 averaging의 단위가 커질수록 LZ 압축한 TIFF 형식의 경우에는 압축률이 낮아지지만, 제안한 방식에서는 압축률이 점진적으로 향상됨을 알 수 있다.

〈표 3〉 Averaging 처리 후 압축 비교
〈Table 3〉 Comparison of compression after averaging

(화일 크기 : 바이트, 압축률 : %)

Averaging 처리 단위	압축전 TIFF 형식	LZ 압축		제안한 방식	
	화일 크기	화일 크기	압축률	화일 크기	압축률
2*2	706362	655257	92.77	498365	70.55
3*3	706362	663933	93.99	479707	67.91
4*4	706362	672665	95.23	463776	65.66
5*5	706362	676431	95.76	450581	63.79
6*6	706362	679107	96.14	439122	62.17
7*7	706362	680637	96.36	429232	60.77
8*8	706362	682757	96.66	422179	59.77



(그림 3) Averaging 처리 후 압축 비교 그래프
(Fig. 3) Comparison graph of compression after averaging

(그림 3)의 그래프에서 위에 나타난 선은 LZ 방식에 의한 압축 수행 결과 화일의 크기다. 제안한 방식에 의한 압축 수행 결과는 그래프의 아래 부분에 나타난 선이다. 화상의 처리 단위를 크게 할수록 LZ 압축 결과의 TIFF 형식 화일 크기는 더욱 커지지만 제안한 방식은 계속 크기가 작아진다. 이것은 제안한 방식의 독성인 데이타의 유사성을 고려한 압축률의 향상을 잘 나타내주는 그래프다.

또다른 예로서 〈표 4〉는 동일한 크기의 화상들에 대하여 TIFF와 TGA 두가지 형식으로 압축을 수행한 결과와 제안한 압축 알고리즘의 수행 결과를 비교한 것이다. 이 화상들은 averaging 처리를 하지 않은 것이다. 실험에 사용한

화상은 대부분 스캐너를 이용하여 입력받은 24 비트 실제 색상 화상이다. 화상의 색상 변화가 거의 없을 때는 기존의 압축 알고리즘이 좋은 성능을 나타낸다. 그러나 실제 색상은 화상의 성격이 계속 변화하므로 기존의 기법에 비하여 본 연구에서 제시한 알고리즘이 좋은 결과를 나타낸다.

(그림 4)는 압축률을 기준으로 〈표 4〉의 내용을 그래프로 표현한 것이다. 평균적으로 제안한 방식이 LZ 압축의 TIFF 형식이나 RLE 압축의 TGA 형식보다 좋은 압축률을 보인다.

〈표 4〉 동일한 크기 화상의 TIFF, TGA, 제안한 방식 실행 결과

〈Table 4〉 Execution results of TIFF, TGA, proposed method for the same sized image

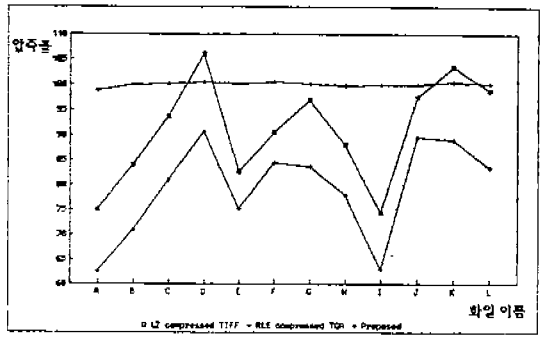
(화일 크기 : 바이트, 압축률 : %)

화일 이름	화일 크기	LZ 압축된 TIFF		RLE 압축된 TGA		제안한 방식	
		화일 크기	압축률	화일 크기	압축률	화일 크기	압축률
A	921612	690579	74.93	912184	98.96	576045	62.50
B	921612	774361	84.02	921122	99.95	653348	70.89
C	921612	863227	93.66	923988	100.26	747110	81.07
D	921612	977491	106.06	926359	100.52	836464	90.76
E	921612	763645	82.86	922467	100.09	692152	75.10
F	921612	833859	90.48	926309	100.51	778078	84.43
G	921612	894045	97.01	923795	100.24	771950	83.76
H	921612	811595	88.06	919743	99.80	719342	78.05
I	921612	685557	74.39	920945	99.93	581039	63.05
J	921612	899709	97.62	922345	100.08	825201	89.54
K	921612	953727	103.48	926223	100.50	820299	89.01
L	921612	910707	98.82	923680	100.22	769345	83.48

〈표 4〉에서 보면 TIFF 형식의 경우 압축률의 변화가 TGA 형식보다 매우 크게 나타남을 볼 수 있는데 이는 (그림 4)에서 더욱 잘 나타난다.

TGA 형식의 경우에는 압축률이 100를 기준으로 적은 편차를 갖는다. 그 이유는 TGA 형식이 RLE 방식의 압축을 수행하거나 혹은 압축을 수행하지 않기 때문이다. 따라서 위와 같이 반복된 데이타가 적은 경우에는 거의 압축을 수행하지 않으므로 이러한 결과가 발생한다. 그러나 TIFF 형식은 LZ 방식의 압축을 수행하므로 압축률의 변화가 TGA 형식보다 크게 나타난다. 〈표 4〉의 시험 화상 중에서 D나 K의 예와 같이 TIFF 형식의 압축 결과가 TGA 형식의 압축 결과보다 좋지 않게 나타나는 경우도 있지만 평균적으로 볼때 TIFF 형식의 압축률이 더 좋다고 할 수 있다. 그리고 제안한 방식의 압축률이 TIFF 형식과 TGA 형식의 압축률보다 높게 나타난다.

(그림 4)의 압축률 비교 그래프에 의하면 대체로 제안한 방식이 기존의 화상 표현 형식과 비교하여 좋은 효율을 나타냄을 명확하게 알 수 있다.



(그림 4) 동일한 크기 화상의 TIFF, TGA, 제안한 방식 실행 결과 그래프

(Fig. 4) Execution results graph of TIFF, TGA, proposed method for the same sized image

5. 결 론

컴퓨터 그래픽의 요구가 다양해짐에 따라 실제 색상 화상의 사용이 증가하고 있다. 이에 따라 기억 용량의 요구가 많은 실제 색상 화상의 압축 문제가 나타난다. 본 연구에서는 실제 색상 화상의 데이타 구조를 변화시키고 데이타 구조의 변경에 따르는 특성을 이용할 수 있는 알고리즘을 개발하여 실제 색상 화상의 효율적인 압축을 수행할 수 있는 방법을 제안하였다. 실험 결과 기존의 화상 표현 형식과의 비교에서 제안한 방식이 평균적으로 높은 압축률을 가짐을 보였다.

본 연구에서 제안한 압축 알고리즘은 데이타의 지역적인 유사성을 이용한 것으로서 데이타의 유사성이 어느 한 점을 기준으로 급변하는 경우에도 압축률에 큰 영향을 미치지 않는다. 즉, 어느 한 부분에서 색상의 변화가 크게 이루어져도 그 이웃하는 픽셀 사이에만 영향을 미친다는 것이다.

기존의 압축 알고리즘들은 데이타가 어느 한 부분을 기준으로 많은 수로 반복하는 경우에 좋은 효율을 나타낸다. LZ 압축의 경우에는 동일한 패턴이 반복되는 경우에 단순히 포인터를 대치하는 것만으로 매우 좋은 압축 효율을 보일 수 있다. 그러나 제안한 압축 알고리즘의 경우에는 인접한 픽셀 간의 유사성만을 이용한다. 따라서 데이타가 큰 단위로 변화하는 경우에는 기존의 방식에 비하여 좋지 않은 압축률을 보일 수도 있다.

24 비트 이상을 이용하는 RGB 데이타로 화상을 표현하는 방식이 사용되는 경우에는 그 소요되는 데이타의 양이 늘어날수록 제안한 알고리즘의 압축률이 더욱 높아지리라는 것을 예상할 수 있다.

추후 연구되어야 될 과제로는 압축시의 실행 시간을 줄일 수 있는 기법에 대한 연구와 압축의 범위를 확장하는 것을 들 수 있다. 래스터화 된 화상은 스캔 라인 단위로 재생 주사되어 하나의 형상을 나타낸다. 그러므로 픽셀 간의 데이타 유사성은 수평적으로만 나타나는 것이 아니라 수직적으로도 나타난다. 제안한 압축 알고리즘에서는 인접한 픽셀의 개념을 한 스캔 라인 상에서 수평적으로만 생각하였다. 인접한 픽셀의 개념을 계속되는 스캔 라인의 범위로 확장하면 복수의 인접 픽셀을 가정할 수 있을 것이다. 이러한 스캔 라인 단위의 압축 기법도 개발할 수 있을 것이다. 또한 현재 2가지의 압축 기법을 사용하기 위한 블럭의 크기와 임계값의 설정이 실험적인 결과에 따라 결정되는데, 이에 대한 정확한 분석이 이루어져야 할 것이다.

참 고 문 헌

[1] Charlse M. Eastman, "Vector versus Raster:A Functional Comparison of Drawing Technologies", IEEE Computer Graphics & Applications, Vol. 10, No. 5, pp. 68-76, SEP., 1990.

[2] Marv Luse, Bitmapped Graphics Programming, pp. 539-549, Addison Wesley, 1993.

[3] Steve Rimmer, Supercharged Bitmapped Graphics, pp. 295-301, McGraw-Hill, 1992.

[4] 홍창완, 박도순, "JPEG을 이용한 이미지 압축의 구현 및 개선", 정보과학회 학술발표 논문집, 제19권, 제2호, pp. 605-608, 1992. 10.

[5] Andy C. Hung, "Image Compression: The Emerging Standard for Color Images", IEEE Computer, Vol. 22, No. 12, pp. 20-27, DEC., 1989.

[6] Marc Nelson, The Data Compression Book, pp. 350-351, Prentice Hall, Inc., 1991.

[7] Timothy Bell, Ian H. Witten, John G. Cleary, "Modelling for Text Compression", ACM Computing Surveys, Vol. 21, No. 4, pp. 576-581, DEC., 1989.

[8] Jacob Ziv, Abraham Lempel, "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-339, MAY, 1977.

[9] 박지환, "Ziv-Lempel 부호와 그 응용에 관한 고찰", 통신정보보호학회지, 제2권, 제2호, pp. 57-61, 1992. 6.

[10] Debra A. Lelewer, Daniel S. Hirschberg, "Data Compression", ACM Computing Surveys, Vol. 10, No. 3, pp. 271-274, SEP., 1987.

송 길 영

1992년 고려대학교 전산과학과 (학사)
1994년 고려대학교 전산과학과 (석사)
1994~현재 한국 무역 정보 통신 연구원
관심분야 : 컴퓨터 그래픽스, EDI시스템, 컴퓨터 통신

조 광 문

1988년 고려대학교 전산과학과 (학사)
1991년 고려대학교 전산과학과 (석사)
1991~현재 고려대학교 전산과학과 박사과정
관심분야 : 이동통신, EDI시스템, 컴퓨터 그래픽스

최 창 원

1990년 고려대학교 전산과학과 (학사)
1992년 고려대학교 전산과학과 (석사)
1992~현재 고려대학교 전산과학과 박사과정
관심분야 : EDI시스템, ISDN, 분산처리 시스템, 컴퓨터 그랙픽스

김 태 윤

1981년 고려대학교 산업공학과 (학사)
1983년 미국 Wayne State University 전산과학과(석사)
1987년 미국 Auburn University 전산과학과(박사)
1988~현재 고려대학교 전산과학과 교수
관심분야 : EDI시스템, ISDN, 이동통신, 위성통신, 컴퓨터 그랙픽스