전처리에 의한 비트 맵 한글 폰트의 압축 방법

우정원*, 김홍배*, 조경연**, 이정현***

* 삼보 컴퓨터 기술 연구소
**부산수산대학교 전자계산학과
***인하대학교 전자계산공학과

Preprocessing Method for the Compression of Bitmapped Hangul Fonts

Jeong Won Woo*, Hong Bae Kim*, Gyung Yun Cho**, Jung Hyun Lee***

*R&D Center, TriGem Computer Inc.
** Dept. of Computer Science, Pusan Su-san University
*** Dept. of Computer Science & Engineering, Inha University

요 약

개인용 컴퓨터의 확산과 함께 사용자 인터페이스도 많은 발전을 하여 비데오 디스플레이의 경우 다양한 서체의 글자에 대한 지원이 필요하게 되었다. 한편 비데오 디스플레이의 경우 사용자의 입력에 대하여 빠른 응답으로써 표현되어야 하므로 복잡한 계산 과정을 갖지 않는 폰트를 이용하여야 한다. 여러 가지 종류의 폰트중에서 이에 적합한 폰트는 비트 맵 폰트이나, 비트 맵 폰트는 그 특성상 모든 종류의 서체에 대하여 각각의 데이타를 따로 가지고 있어야 하므로 저장 매체의 용량이 많이 필요하다. 그러므로 이에 대하여 비트 맵 폰트를 압축하여 저장한 다음 사용시에 하드웨어에 의하여 빠르게 복원함으로써 사용자의 입력에 대하여 빠른 응답으로 대처하는 방법에 대한 연구가 이루어지고 있다. 본 논문에서는 간단한 하드웨어를 이용하여 압축 이전에 폰트를 전처리함으로서 기존의 압축을 개선하기 위한 전처리 방법을 제안한다.

1. 서 론

데이타 압축은 단순히 데이타의 효율적인 저장뿐만 아니라 전송 선로를 통한 데이타의 전송에 있어서도 빠른 시간 내에 효율적으로 데이타를 전송할 수 있게 한다[1-3]. 압축 방법은 압축될 데이타의 형태 및 그 압축된 데이타가 사용되는 응용에 따라 결정된다[4]. 또한, 압축 효율을 높이기 위하여 많은 응용에서 압축 이전에 전처리의 기법을 사용하고 있다[5-7]. 압축과 복원에 있어 구현을 하드웨어로 할 경우, 하드웨어의 복잡도가 응용 가능토록 구성하는 것이 문제시 되고 있다[8-10]. 한편, 개인용 컴퓨터의 확산과 함께 사용자 인터페이스도 많은 발전을 하여 비데오 디스플레이에 있어 다양한 글자꼴의 지원이 필요하게 되었다. 여러 글자꼴을 비트 맵 방식의 폰트를 가지고 지원하기 위해서는 각 글자꼴에 대한 폰트 데이타를 따로 갖고 있어야 하기 때문에 많은 양의 저장 장치를 필요로 한다. 한글의 경우 영문에 비하여 글자수가 많기 때문에, 각 서체의 데이타 양이 영어 문화권의 것에 비해 훨씬 많다. 그러므로 비트 맵 폰트를 이용하여 한글의 여러 서체를 지원하기 위해서는 비트 맵 폰트 데이타의 크기를 줄이기 위한 압축 방법이 필요하다. 또한, 비데오 디스플레이의 특성상 빠른 I/O에 대응하여야 하므로 압축된 데이타에 대한 복원은 하드웨어로 구현하는 것이 필요하다. 한편 하드웨어로 구현함에 있어서, 구현이 비용적으로 가능한 수준이 되어야 하므로 하드웨어 복잡도를 줄여야 한다. 이에 따라, 최근에는 복원시의 하드웨어 복잡도를 사용 가능토록 줄이기 위하여 압축률의 저하를 감수하면서도 H/W 복잡도를 낮추는 방법이 제안되었다[15].

본 논문에서는 간단한 하드웨어로 복원이 가능한 전처리 방법을 비트 맵 폰트에 적용하여 압축률을 높이는 방법을 제안한다.

논문의 구성은 2장에서는 기존의 비트 맵 폰트의 압축 방법에 대하여 기술하였으며, 3장에서는 비트 맵 한글 폰트 압축을 위한 전처리 방법들과 이들의 복원시 필요한 하드웨어의 구성을 제시하였고, 4장에서는 이를 기존에 제안된 압축 방법에 적용하여 그 결과들을 분석 하였고, 5장에서는 결론을 기술하였다.

2. 기존의 비트 맵 폰트의 압축 방법

2.1 비데오 디스플레이를 위한 비트 맵 폰트의 압축

압축은 구현 방법에 있어 소프트웨어적 방법과 하드웨어적 방법으로 나뉘어 진다. 소프트웨어적 방법은 구현에 있어서 별도의 하드웨어를 필요로 하지 않아 하드웨어에 무관하게 구현할 수 있다는 장점이 있으나 실행 속도면에서 뒤떨어지기 때문에 빠른 I/O 처리를 요구하는 응용에는 적용이 어렵다[12]. 하드웨어를 이용한 압축 방법은 실행 속도면에서는 소프트웨어를 이용한 방법보다 월등하지만, 응용에 맞는 각각의 하드웨어를 구현해야 하기 때문에 많은 비용과 시간이 소요된다. 그리고 그 비용과 시간은 하드웨어의 복잡도에 따라 그 정도가 다르다.

한편 압축 방법에는 압축 및 복원 과정에서 원래의 데이타에 대하여 손실을 갖는 비가역 압축과 원래의 데이타를 그대로 유지하는 가역 압축이 있다. 일반 화상이나 동영상과 같은 부류의 정보는 그 응용에 따라 비가역 압축 방법을 사용하여도 사람의 눈으로 인지하기가 어렵다. 그러나 비데오 디스플레이의 경우 조그마한 폰트의 변형에 대하여도 사용자가 그 변형을 인지하기가 쉽기 때문에 가역 압축 방법을 사용해야 한다.

2.2 런-렝스 허프만 부호화 (RHC)

이 방법은 런-렝스 부호화와 허프만 부호화를 결합한 방법으로 비트 맵 데이타를 검은점('1') 및 흰점('0')의 연속수에 의한 길이로 표현함에 있어서 그 발생 빈도에 의하여 코드를 부여하는 방법이다. 이 방법은 간단히 코드 테이블만을 가지고 복원이 가능하기 때문에 하드웨어적으로 구현이 가능하여 현재 많은 응용 분야에서 사용되고 있다 [13,14]. 그러나 코드의 종류가 많을 경우 코드 테이블이 커지는 단점이 있다.

2.3 변형 허프만 부호화 (MHC)

런-렝스 허프만 부호화에 있어서 변환된 데이타를 복원하기 위해서는 런-렝스 종류 만큼의 테이블이 필요하다. 따라서 런-렝스의 종류가 극단적으로 많을 경우 복원을 위한 하드웨어는 매우 복잡하고, 복원시 시간도 오래 걸린다. 변형 허프만 부호화는 이러한 단점을 극복하기 위하여 제안된 방법으로 런-렝스에 있어서 일정한 배수마다 Make-up 코드와 그 배수 이하의 수를 나타내기 위한 Terminating 코드를 가지고 허프만 코드를 구현하고 있다.

2.4 변형된 변형 허프만 부호화 (MMHC)

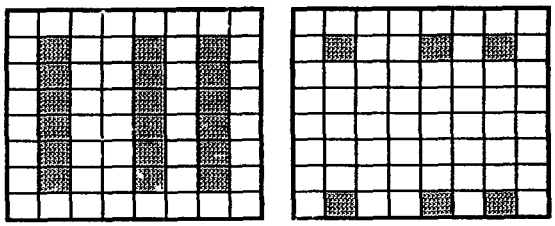
허프만 코드와 같이 테이블에 근거하여 압축된 데이타를 복원함에 있어, 그 구현을 하드웨어로 할 경우 복원기의 성능과 하드웨어의 복잡도, 유연성 및 테이블의 크기 사이에는 상관 관계가 존재한다 [11,12,13]. 변형된 변형 허프만 부호화는 코드의 수를 줄임으로서 하드웨어의 구성을 간단하게 하기 위한 방법으로서 최근에 제안되었다. 이 방법은 변형 허프만 부호화를 변형한 것으로 변형 허프만 부호화가 Make-up 코드를 일정 수의 배수 (8 배수, 16 배수, 32 배수...)들로 구성하는 것에 비해 이를 2^n 배수들로 구성 함으로서 Make-up 코드의 종류를 줄이고 있다[15]. 그러나 이 방법은 하드웨어의 복잡도는 많이 개선하였으나 폰트 데이타에 대한 압축률이 감소하는 단점이 있다.

3. 전처리 방법의 설계

비트 맵 폰트 데이타를 런-렝스에 의한 허프만 부호로 부호화 할 경우 이 부호화된 데이타의 크기는 전체 코드 수, 코드의 종류 수 및 이들 코드들 간에 있어서 데이타의 분포 형태에 의하여 영향을 받는다. 한편, 비트 맵 폰트 데이타는 그 특성상 데이타들이 횡축이나 종축 혹은 대각선 형태로 일정개 연결되는 특성을 갖는다. 본 논문에서는 압축률에 영향을 주는 요소를 변환하여 압축률을 높이기 위한 방법으로써 데이타를 전처리 하는 방법에 대하여 제안한다. 이 전처리 방법들은 폰트 내의 획의 특성 및 비트 맵 데이타의 행간 관계를 이용하는 방법으로서 다음과 같이 세 가지 방법을 사용하였다.

3.1 행간 XORing에 의한 전처리

행간 XORing은 행간의 각 데이타 쌍에 대하여 두개의 데이터가 같은 경우는 '0'으로, 다른 경우는 '1'로 변경하는 방법이다. 이는 글자 획의 유형 중 종축으로 점이 연결된 경우에 우선적으로 적용을 하기 위한 방법으로 압축률에 영향을 주는 요소인 전체 부호 수를 줄여 압축률을 높이는 방법이다. 변환 예를 [그림 1]에 나타내었다.



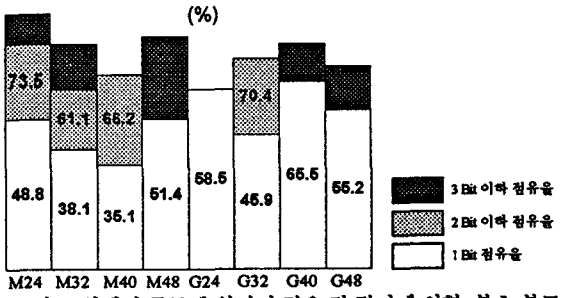
(a) 패턴 예 (b) (a)에 대한 행간 XORing 후

[그림 1] 행간 XORing 예

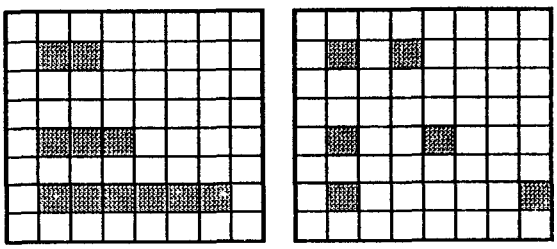
3.2 열간 XORing에 의한 전처리

허프만 부호의 특성상 비록 같은 수의 허프만 부호를 갖고 있더라도 그 데이타들의 분포 형태에 따라 전체 데이타의 크기가 다르게 된다. 8개의 한글 폰트에 대한 허프만 부호화 결과를 살펴 보면 검은점에 대한 허프만 부호의 분포 형태는 [그림 2]와 같이 1 비트의 허프만 부호의 점유율이 약 50% 정도이고 2 비트까지의 점유율은 63%, 3 비트까지의 점유율이 78%로 나타난다. 그러므로 이 분포 형태를 가능한 적은 비트의 점유율을 크게(3 비트 이하의 점유율이 99% 이상 되도록) 변환할 수 있다면 압축률을 향상시킬 수 있다. 열간 XORing

방법은 위와 같이 널리 분포되어 있는 코드들을 몇 개의 코드에 집중시키는 방법이다. 본 논문에서는 해당 도트 데이타를 바로 앞 열의 데이타와 XORing을 하는 방법으로 이를 구현하였다. 변환 예를 [그림 3]에 나타내었다.



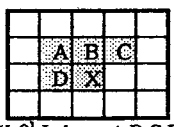
[그림 2] 원래의 폰트에 있어서 검은 점 길이에 의한 부호 분포


(a) 패턴 예 (b) (a)에 대한 열간 XORing 후

[그림 3] 열간 XORing 예

3.3 주위 데이타를 이용한 예상에 의한 전처리

이 방법은 각 도트의 값을 주위의 데이타 값에 의하여 예상함으로써 흑과 백의 데이타 변화를 줄여 전체 코드의 수 및 코드의 분포를 변화시켜 압축률을 향상시키는 방법이다. 예상은 아래의 [그림 4]와 같이 결정하고자 하는 도트 주변 4개의 도트 데이타를 이용하였다. 이 변환은 2-패스에 의하여 이루어진다. 첫번째 패스에서는 예상 테이블 내의 값을 결정하기 위한 데이타를 수집하고 이에 의한 테이블을 구성한다. 두번째 패스에서는 첫번째 패스에서 결정된 예상 테이블과 해당 폰트를 비교하여 폰트 데이타를 재구성한다. 이때, 해당 데이타가 예상 테이블과 일치하는 경우는 '0'으로 일치하지 않는 경우는 '1'로 변환한다.

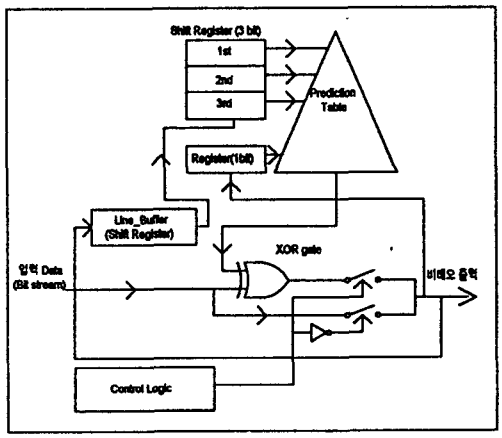


X 의 Index = A,B,C,D

[그림 4] 예상을 위한 주변 도트(Dot) 데이타

3.4 전처리를 위한 하드웨어의 구성

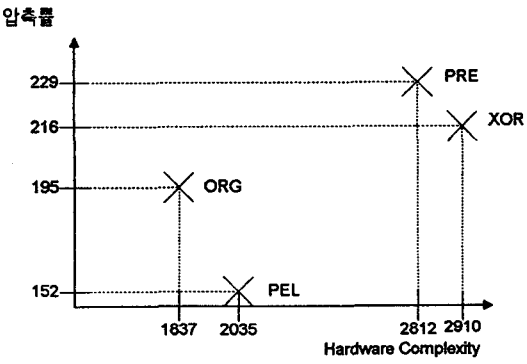
[그림 5]에 예상에 의한 전처리 복원 하드웨어의 구성을 나타내었다. 구성은 크게 예상 테이블을 가지고 해당 값을 구하기 위한 부분(상부)과 해당 데이타의 윗 행과 전 열에 대한 데이타의 저장 및 이에 대한 XORing을 위한 부분(하부) 으로 구성된다. 행간 XORing이나 열간 XORing에 대한 복원 회로는 [그림 5]의 하부 구조 만으로 구현이 가능하다. 그러므로 하드웨어 복잡도에 있어서는 예상에 의한 전처리에 대한 복원 하드웨어가 가장 복잡하나 비데오 시스템 전체에 대하여 비교하여 볼때 이는 거의 무시할 수 있는 정도이다.



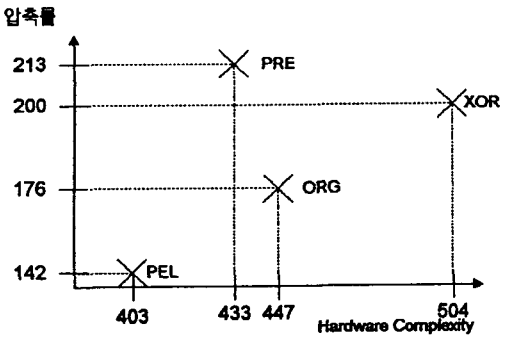
[그림 5] 예상에 의한 전처리에 대한 복원 하드웨어 구성

4. 실험 및 평가

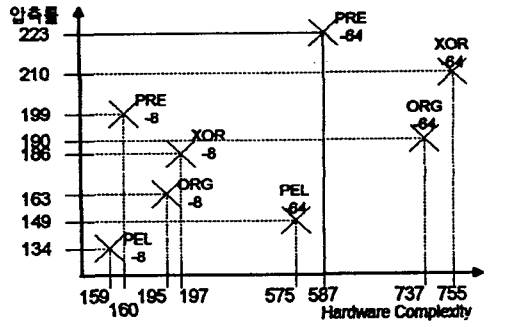
이 장에서는 3장에서 제시한 3가지의 전처리 방법을 KS 완성형 한글(2350자) 고딕체와 명조체 각각 24 X 24, 32 X 32, 40 X 40, 48 X 48 폰트에 적용한 후, 기존의 압축 방법인 런-렝스 허프만 부호, 변형 허프만 부호, 변형된 변형 허프만 부호를 적용하여 각각의 압축률의 개선을 비교하였다. 각각의 부호화에 대한 결과를 [그림 6]- [그림 8]에 나타내었다.



[그림 6] 허프만 부호 적용시 압축률 및 하드웨어 복잡도



[그림 7] 16배수 변형 허프만 부호 적용시 압축률 및 하드웨어 복잡도



[그림 8] 8배수 및 64배수의 변형된 변형 허프만 부호 적용시 압축률 및 H/W 복잡도 비교

실험 결과를 보면 열간 XORing의 경우는 오히려 전체 코드의 수가 증가하여 압축률이 떨어졌으나 예상과 행간 XORing에 의한 전처리는 어떠한 방법의 부호화를 사용하여도 압축률이 많이 개선이 되었다.

5. 결 론

비트 맵 폰트의 압축에 있어서 압축 이전에 행간 XORing이나 예상에 의한 전처리 변환을 할 경우 압축률이 많이 개선되었다. 압축 방법중 기존의 8배수에 의한 변형된 변형 허프만 부호가 가장 간단한 복원 하드웨어를 갖으나 압축률은 런-렝스 허프만 부호화 보다 떨어졌다. 그러나 이 방법에 본 논문에서 제안한 전처리 방법들을 적용한 결과 원래의 압축률보다 좋은 압축률을 얻을 수 있었다. 즉, 간단한 복원 하드웨어를 위하여 런-렝스 방법 대신에 변형된 변형 허프만 부호를 사용시 압축률의 저하를 감수 하여야 하는 것에 대하여 전처리 방법을 적용함으로써 기존의 압축률 이상을 유지 할 수 있다.

참고 문헌

[1] Ioannis Pitas, *Digital Image Processing Algorithms*, Prentice Hall, 1993.

[2] Fred Halsall, *Data Communications, Computer Networks and Open Systems*, Addison-Wesley Publishing Company, 1992.

[3] Edward R.Fiala and Daniel H.Greene, "Data Compression with finite Windows" Comm. of the ACM, vol.32, no.4, pp.490-505, April 1989.

[4] Glen Searfoss, "Bounding Box Data Compression," Dr.Dobb's Journal, pp.56-64, April 1990.

[5] D.L.Gall, "A Video Compression Standard for Multimedia Applications," Comm. of the ACM, vol.34, no.4, pp.47-58, April 1991.

[6] F.Sijstermans, J.Meer, "CD-I Full-Motion Video Encoding on a Parallel Computer," Comm. of the ACM, vol.34, no.4, pp.82-91, April 1991.

[7] G.K.Wallace, "The JPEG Still Picture Compression Standard," Comm.of the ACM, vol.34, no.4, pp.32-44, April 1991.

[8] D.Royals, T.Markas, N.Kanopoulos, J.H.Reif, J.A.Storer, "On the Design and Implementation of a Lossless Data Compression and Decompression Chip," IEEE Journal of Solid-State Circuits, vol.28, no.9, Sep. 1993.

[9] LA.Shah, O.A.Assani,B.Johnson, "A Chip Set for Lossless Image Compression," IEEE Journal of Solid-State Circuits, vol.26, no.3, pp.237-244, Mar.1991.

[10] H.Park, V.K.Prasanna, "Area Efficient VLSI Architectures for Huffman Coding," IEEE Trans. on Circuits and Systems, vol.40, no.9, pp.568-575, Sep. 1993.

[11] M.T.Sun, T.C.Chen and Albert M.Gottlieb, "VLSI Implementation of a 16 x 16 Discrete Cosine transform," IEEE Trans. on Circuits and Systems , vol.36 , pp.610-617, April 1989.

[12] Amar Mukherjee, N.Ranganathan, "Efficient VLSI Designs for Data transformation of Tree-Based Codes," IEEE Trans. on Circuites and Systems, Vol.38, No.3. pp.306-314 , Mar. 1991.

[13] Keshab K . Parhi , "High - Speed VLSI Architectures for Huffman and Viterbi Decoders," IEEE Trans. on Circuits and Systems , vol.39 , No.6, pp.948-953, June 1992.

[14] S.Golomb, "Run-length Encodings," IEEE Tran. Inform. Theory, vol. IT-12, pp.399-401, July 1966.

[15] 김종북, "한글 폰트의 압축과 복원에 관한 연구," 홍익대 이학석사학위논문, 1994.