한국산학기술학회논문지
Vol. 8, No. 6, pp. 1419-1427, 2007

윤곽선 맵과 다중 면 사이드 매치 유한상태 벡터 양자화를 이용한 영상 압축

조성환[1*], 김응성[2]

Image Compression Using Edge Map And Multi-Sided Side Match Finite-State Vector Quantization

Seong-Hwan Cho[1*] and Eung-Sung Kim[2]

요 약 본 논문에서는 영상의 윤곽선을 검출하여 배경 블록과 윤곽선 블록으로 분류하고 윤곽선 맵을 작성하여, 윤곽선 블록에 대해서는 다시 DCT의 AC 계수를 사용하여 16개로 세분화한 후, 다중 면 사이드 매치 유한상태 벡터 양자화를 수행하는 알고리듬을 제안한다. 윤곽선 맵의 정보에 따라 각각 주 부호책으로부터 상태 부호책을 작성하며, 현재 블록의 2면 또는 3면에 대해 사이드 매치 계산을 수행한다. 전송 비트 수를 줄이기 위해 먼저 부호화되는 블록들 중 배경 블록에 한하여 주 부호책으로 부호화 할 것인지를 결정한다. 또한 복호화기로 전송하는 부호단어 인덱스의 할당 비트를 줄이기 위해서 가변 길이 부호화를 수행한다. Zelda, Lenna, Bridge, Peppers 영상에 대하여 본 알고리듬으로 영상을 부호화 했을 때 SMVQ와 TSMVQ 알고리듬보다 더 좋은 영상의 화질을 얻을 수 있었다.

Abstract In this paper, we propose an algorithm which implements a multi-sided side match finite-state vector quantization(MSMVQ). After extracting the edge information from an image and classifying the image into edge blocks or non-edge blocks, we construct an edge map. We subdivide edge blocks into sixteen classes using discrete cosine transform(DCT) AC coefficients. Based on edge map information, a state codebook is made from the master codebook, and side match calculation is done for two-sided or three-sided current block of image. For reducing transmitted bits, a decision is made whether or not to encode the non-edge blocks among the pre-coded blocks by using the master codebook. Also for reducing allocation bits of codeword indices to decoder, a variable length coder is used. Considering the comparison with side match finite-state vector quantization(SMVQ) and two-sided SMVQ(TSMVQ) algorithm about Zelda, Lenna, Bridge and Peppers image, the new algorithm shows better picture quality than SMVQ and TSMVQ respectively.

Key Words : Vector Quantization, SMVQ, TSMVQ, MSMVQ

1. 서론

디지털 영상 데이터를 전송하거나 저장하기 위해서는 전송채널의 대역폭이 커야 되며 큰 용량의 메모리가 필요하다. 그러므로 저장하거나 전송할 영상 데이터의 크기를 줄여 주기 위해 영상 압축(image compression)이라고 하는 처리 과정을 거치게 된다. 여러 가지 영상 데이터 압축 기술 중에서 벡터 양자화(Vector Quantization : VQ)[1]는 적은 비트율, 즉 영상의 화소당 1 비트 이하로 영상을 부호화하는 효과적인 공간 영역 영상 부호화 방법이다. 간단한 벡터 양자화는 k 차원의 유클리디언 공간 R^k의 Q를 R^k의 유한 부분 집합 Y로의 맵핑(mapping)으로 정의할 수 있다. 즉 $Q: R^k \rightarrow Y$로 쓸 수 있는데 $Y=\{Y_i ; i=1,2,\ldots,N\}$는 재구성 벡터 집합이고 N은 Y에서 벡터의 수이다. 이때 부분 집합 Y를 부호책(codebook)이라 부른다. k 화소들의 하나의 영상 블록 X는 부호책에 있는 부호단어(codeword)라고 하는 대표 벡터에 의해 표현된다. 단지 부호단어의 주소만이

이 논문은 2006년도 금강대학교 교내연구비의 지원에 의하여 연구되었음
[1]금강대학교 교양학부
[2]경기공업대학 컴퓨터정보시스템과
*교신저자: 조성환(shcho@ggu.ac.kr)

전송되거나 저장되기 때문에 부호책 크기가 N인 VQ를 이용한 영상 부호화의 비트율은 $(\log_2 N)/k$로 쓸 수 있다.

복원 영상의 화질을 좋게 하고 효과적인 압축 기법을 위해서 벡터 양자화에 대한 여러 가지 알고리듬들이 개발되고 있다[1-3]. 이들 알고리듬에는 블록의 크기를 변화시켜 각각에 대해 양자화를 수행하는 가변 블록 벡터 양자화[4]나 다른 화소 값 변화의 특징을 가지는 블록을 다른 계층(layer)에 할당시키는 계층적 다중비율 벡터 양자화(Hierarchical Multirate VQ : HMVQ)[5]등이 있는데, 이들 알고리듬에서는 영상의 특징들을 강조함으로써 $(\log_2 N)/k$의 k 값을 증가시켜 비트율을 감소시킨다. 또한 부호단어에 할당하는 비트를 감소시키는 알고리듬[6]도 연구되었는데, 즉, 대표 벡터 선택을 위한 부호책의 부호단어 수를 조절하여 비트율 $(\log_2 N)/k$에서 $(\log_2 N)$의 값을 감소시킴으로써 전체 비트율을 감소시킨다.

또한 블록 경계를 가로지르는 이웃 화소들의 상관관계(correlation)를 이용함으로써 최적 대표 벡터를 찾기 위해 이전에 부호화된 블록들을 사용하는 유한상태 벡터 양자화(Finite-State Vector Quantization : FSVQ)[7-8] 알고리듬이 있는데, 이와 같은 이웃 블록간의 상관관계를 이용함으로써 일반적인 벡터 양자화기로 가능한 것 이하로 비트율을 감소시킬 수 있다.

H. Wei와 P. Tsai, J. Wang은 기존의 사이드 매치(side match) 유한상태 벡터 양자화(SMVQ)에서 사용하는 2면(two-sided)을 사용한 방법이 아닌 3면(three-sided) SMVQ(TSMVQ)를 제안하였다[9]. 제안한 방법을 사용하면 현재 블록의 좌측과 상위 블록만을 사용하여 양자화를 하는 경우 발생하는 오류를 줄이게 된다.

또한 프랙탈(Fractal)을 사용하여 사이드 매치 벡터 양자화를 수행하는 기법이 Hsuan T. Chang에 의해 제안되었는데[10], 기존의 프랙탈 블록 부호화와는 달리 주 부호책(master codebook)으로부터 동적으로 상태 부호책을 추출하여 사이드 매치와 그래디언트(gradient) 매치를 동시에 수행하여 비트율을 줄이게 된다. 최근에는 이러한 사이드 매치 기법을 동영상 부호화에 적용한 논문도 발표되었다[11, 12].

본 논문에서는 정지 영상의 특징 판별을 확실히 하고 전송 비트율을 낮추기 위하여 윤곽선을 검출하여 배경 블록과 윤곽선 블록으로 분류한 윤곽선 맵(edge map)을 작성하고, DCT(Discrete Cosine Transform)의 AC 계수를 사용하여 윤곽선 영상 블록을 16개로 세분화한 후, 다중면(multi-sided) SMVQ를 수행하는 알고리듬을 제안한다. 전송 비트 수를 줄이기 위해서 2면이나 3면 SMVQ를 수행 전에 먼저 부호화되는 블록들 중 배경 블록에 한하여 주 부호책으로 부호화 할 것인지를 결정하는 알고리듬을 제안하며, 전송 비트 수를 더 줄이기 위해 복호화기로 전송되는 부호단어 인덱스에 대해서 가변 길이 부호를 할당하는 알고리듬을 사용한다. 시뮬레이션 결과, 본 논문에서 제안한 방법으로 영상 부호화를 했을 경우 기존의 벡터 양자화 기법인 SMVQ[8]나 TSMVQ[9]와 비교하여 더 좋은 화질의 영상을 얻을 수 있었다.

2. 윤곽선 맵 작성

벡터 양자화 기법은 일반적으로 윤곽선 근처에서 고주파수 양자화 오류를 일으키고, 배경 부분에서는 저주파수 양자화 오류를 발생시킨다. 날카로운 윤곽선 부분의 주파수 오류는 윤곽선을 무디게 만들며 이는 복원 영상의 화질에 큰 영향을 준다. 배경 영역에서의 왜곡(distortion)은 블록화 현상(blocking effect)이 없다면 인간의 눈에 명백하게 관찰되지 않는다.

본 논문에서는 이와 같은 일반적인 벡터 양자화 기법에서 발생하는 오류를 최소화하기 위해 블록들을 배경 블록과 윤곽선 블록으로 분류한 후, 윤곽선 맵을 작성하여 벡터 양자화를 수행하는 알고리듬을 제안한다. 본 논문에서 제안하는 유한상태 분류벡터 양자화기를 사용하기 위해서는 벡터 양자화를 위한 부호책이 두 가지, 즉 주 부호책과 상태 부호책(state codebook)이 존재해야 한다. 주 부호책은 입력 영상의 윤곽선 정보를 이용하여 블록을 배경 부분과 16개의 윤곽선 부분으로 분류하여 각 분류 클래스에 따라 따로 작성한다. 각각의 주 부호책은 LBG 알고리듬[13]을 이용하여 배경 주 부호책과 16개의 윤곽선 주 부호책을 작성하는데, 입력 영상의 한 블록에 대해 그 블록이 배경 블록인지 윤곽선 블록인지를 결정하기 위하여 입력 영상에 대해 윤곽선 검출을 해야 한다. 하나의 입력 영상 전체에 대하여 Sobel 연산자를 이용하여 윤곽선을 검출한다. 윤곽선 검출이 끝난 후, 입력 영상을 부 블록(sub block)으로 나누어 각 블록에 대해 윤곽선이 존재하는지 여부를 결정한다. 입력 영상의 윤곽선 검출 결과에 대해 [그림 1]과 같이 윤곽선 영상을 4×4 블록으로 나누어 그 4×4 블록내의 16개 화소 값에 대하여 문턱치(threshold)를 주어 그 중에 하나라도 문턱치보다 큰 값을 가진 화소가 존재하면 그 블록은 윤곽선 블록으로 간주한다. 일반적으로 윤곽선 검출된 영상의 화소 값은 밝은 값 아니면 어두운 값을 가지게 되므로 윤곽선 블록 검출에 사용한 문턱치는 일반 영상의 화소 값 범위(0~255)의 반인 128을 사용하였다. 윤곽선 블록에 따라서 윤

곽선 정보를 가지고 있는 맵(map)을 작성하는데, 윤곽선 맵의 하나의 위치는 입력 영상의 한 블록 4×4에 해당한다. [그림 1]에서와 같이 현재 입력 블록이 윤곽선 블록이면 해당 윤곽선 맵의 값을 1로, 배경 블록이면 맵의 값을 0으로 한다.

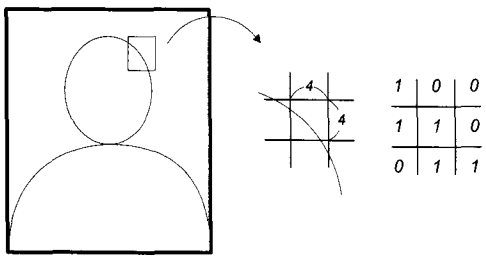


그림 1. 윤곽선 맵 작성

3. 윤곽선 블록의 16 클래스 분류

배경 블록에 비해 윤곽선 블록들은 영상의 중요한 요소인 윤곽선 정보를 가지고 있기 때문에 모든 윤곽선 블록들을 그대로 하나의 주 부호책 설계에 사용하면 여러 가지의 윤곽선 정보가 부호책에 반영되지 않는다. 그러므로 배경 블록에 대해서는 그 블록들을 모아 바로 주 부호책 설계에 입력 벡터로 들어가고 윤곽선 맵의 값이 1인 윤곽선 블록들은 다시 16개의 서브 클래스로 세분화한다. 영상의 시각적인 특징을 유지하는 벡터 양자화기의 한 종류로서 Ramamurthi와 Gersho에 의해 분류 벡터 양자화(CVQ)가 제안되었다[14]. 분류 벡터 양자화에서는 윤곽선의 방향, 위치 등에 의해서나 배경의 밝기 영역에 따라 각 블록을 분류한다. 본 논문에서는 입력 벡터의 분류를 위해서 영상의 시각정보를 잘 유지하는 DCT(Discrete Cosine Transform)를 사용하였다. 각 윤곽선 블록 벡터에 대한 16 클래스로의 분류는 현재 블록에 대한 DCT를 계산하여 그 계수 값의 성질을 이용한다.

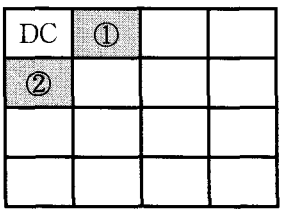


그림 2. 윤곽선의 16 클래스 분류를 위한 DCT 계수값 선택

윤곽선 맵의 값이 1인 윤곽선 블록의 원 영상 화소 값에 대해 DCT 계수 값을 계산하고, 그 결과 [그림 2]에서와 같이 ①, ②번의 AC 계수만을 취하여 윤곽선의 종류를 판별하게 된다[15].

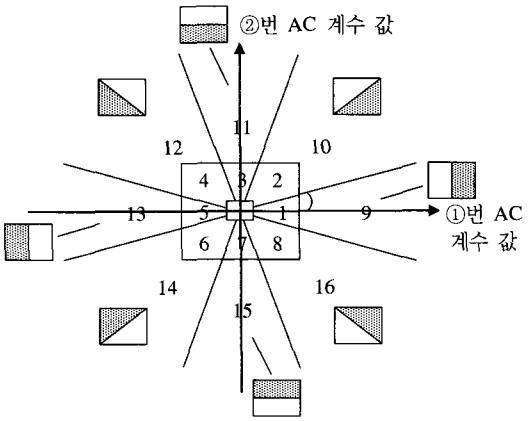


그림 3. AC 계수 값에 의한 윤곽선의 형태

이와 같이 두 AC 값을 [그림 3]과 같은 x, y 좌표축으로 맵핑시켜 현재 블록이 각 16개의 클래스 중에 어느 영역에 속하는지를 판단하여 윤곽선의 종류를 분류한다. [그림 3]에서 16개의 영역이 존재하는데 그림에서와 같이 각 영역은 윤곽선의 성질을 반영하고 있으며, 그림의 원점 근처 작은 사각형은 원래는 배경 블록이지만, Sobel 후에 4×4 블록내의 16개 윤곽선 화소 값에 대하여 하나라도 문턱치보다 큰 값을 가진 화소가 존재하여 윤곽선 블록으로 분류된 블록에 대해 두 AC 계수 값들이 0에 근접하는 성질을 이용함으로써 다시 배경 블록으로 수정하기 위해 설정해 놓은 것이다. 그림내의 바깥쪽 사각형을 기준으로 바깥쪽 영역은 윤곽선 블록내의 밝기 차이가 상당히 큰 부분이고 안쪽 영역은 비교적 윤곽선의 밝기 차이가 덜 한 부분이다. 이와 같이 윤곽선 블록들에 대해 윤곽선 유형에 따라 16개의 클래스 중 하나로 분류하고, 윤곽선 맵에 그 클래스 종류를 저장한다. 만약 현재 블록이 배경 블록이면 '0'(1비트)을 윤곽선 맵에 저장하며, 윤곽선 블록이라면 '1' 뒤에 윤곽선 종류 값(1~16)을 4비트(총 5비트)로 하여 맵에 저장한다.

윤곽선을 변경한 후에, 배경 블록들은 배경 블록대로, 16개의 윤곽선 블록들도 각 클래스 별로 따로 모아 주 부호책을 작성한다. 배경 블록과 16개 클래스의 윤곽선 블록들에 대해 주 부호책을 작성하면 총 17개의 주 부호책이 작성된다.

4. 다중 면 사이드 매치 유한상태 벡터 양자화기

기존의 SMVQ[8]는 좌측과 상위 블록의 각각 4개의 화소만을 사용하여 현재 블록의 상태 부호책 인덱스를 결정한다. H. Wei와 P. Tsai, J. Wang은 이와 같이 2면(two-sided)만 사용하여 상태 부호책을 결정하는 문제점을 해결하기 위해 3면(three-sided) SMVQ(TSMVQ)를 제안하였다[9]. 현재 블록의 주변 3개 블록의 면을 사용하여 왜곡을 계산하면 2개 블록의 2면을 사용할 때보다 영상의 화질은 좋아지지만, 많은 수의 블록에 대해 SMVQ를 사용하지 않고 기존 벡터 양자화기를 사용함으로써 전송 비트 량이 많아지게 된다. 본 논문에서는 기존의 2면 또는 3면과 4면을 사용하여 SMVQ를 실행하면서도 전송 비트 량을 줄일 수 있는 다중 면(multi-sided) SMVQ(MSMVQ) 알고리듬을 제안한다.

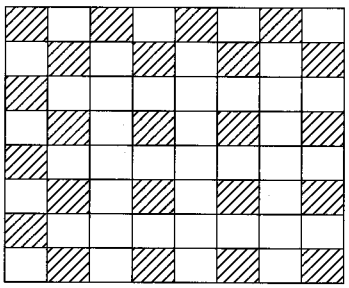


그림 4. 주 부호책 사용 벡터 양자화 블록

먼저 기존 SMVQ[8]에서 사용하는 2면의 유한상태 벡터 양자화가 아닌 현재 블록의 3면을 이용한 사이드 매치를 계산을 위해 4×4 블록 중에서 상태 부호책이 아닌 주 부호책을 사용하여 벡터 양자화를 수행하게 한다. [그림 4]에서 ▨로 표시된 4×4 블록들은 SMVQ를 사용하지 않고, 주 부호책을 사용하여 LBG 알고리듬으로 □ 블록들보다 먼저 벡터 양자화를 실행하는 블록들을 나타낸다.

▨ 블록들에 대해 주 부호책을 사용하여 벡터 양자화를 수행하면 상태 부호책을 사용할 때보다 그 만큼 전송 비트 수가 많아지게 된다. 그래서 무조건 ▨ 블록들에 대해 벡터 양자화를 실행하지 않고, 전송 비트 량을 줄이기 위해 [그림 5]과 같은 알고리듬을 적용하여 ▨ 블록들에 대해 주 부호책을 사용한 벡터 양자화를 수행할지 여부를 결정하게 된다.

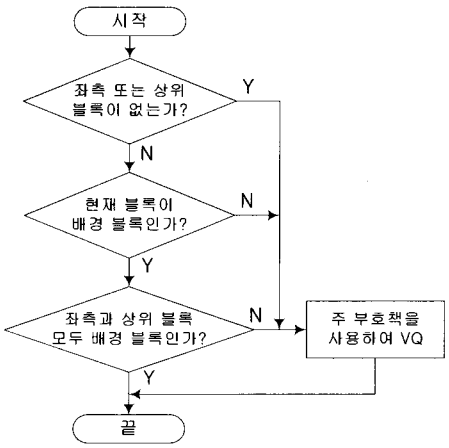


그림 5. ▨ 블록들에 대한 주 부호책 벡터 양자화 결정 알고리듬

▨ 블록들만을 대상으로 [그림 5]의 알고리듬을 적용하여 먼저 벡터 양자화를 할지를 결정하는 이유는 현재

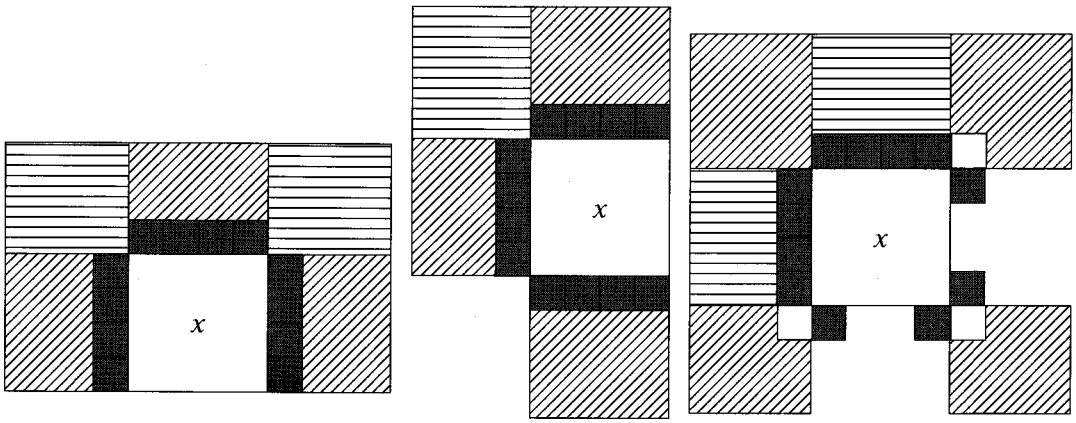


그림 6. MSMVQ에 대한 3가지 사이드 매치 유형

블록이 배경 블록이고 좌측과 상위 블록이 모두 배경 블록이라면 굳이 현재 블록을 주 부호책을 사용하여 벡터 양자화를 할 필요가 없기 때문이다. 만약 주 부호책이 256의 크기를 갖는다면 주 부호책을 이용하여 벡터 양자화를 할 경우, 8비트가 소요되지만, 현재 블록이 배경 블록이라면 5절에서 설명하는 방법을 이용하면 전송 비트가 없게 된다. 그러므로 현재의 ▨ 블록이 배경 블록일 확률이 높은 경우 벡터 양자화를 하지 않는다면 윤곽선 맵의 사용과 주 부호책으로 벡터 양자화를 함으로써 증가되는 전송 비트 수를 줄일 수 있다.

[그림 6]은 MSMVQ를 사용하여 벡터 양자화를 하는 경우, ▨ 블록을 먼저 벡터 양자화 했을 때, 현재 입력 벡터 x에 대해 사이드 매치 계산을 할 수 있는 3가지 유형을 보여 주고 있다. 그림에서 ▨ 블록은 주 부호책을 이용하여 먼저 벡터 양자화된 블록이고, ▤ 블록은 현재 입력 벡터 x보다 먼저 양자화된 블록을 의미한다. 그림에서 보듯이 (a)와 (b)는 2면이 아닌 3면을 이용하여 사이드 매치 벡터 양자화를 수행함으로써 보다 더 정확히 벡터 양자화를 수행할 수 있으며, 그림 (c)의 경우는 입력 벡터 x의 주위로 좌측과 상위의 ▤ 블록만을 사용하여 2면 사이드 매치 벡터 양자화만 수행하게 된다. 이러한 단점을 극복하기 위해 먼저 벡터 양자화된 좌측 하단, 우측 하단, 우측 상단의 ▨ 블록에서 □ 화소 값을 사용하여 입력 벡터 x에 대해 사이드 매치 벡터 양자화를 수행한다. 즉, 완전하게 4면을 사용하는 것은 아니지만, 이와 같은 방법을 사용하면 현재 블록 x의 하단과 우측에 대해 두 개 화소씩 사이드 매치 계산을 할 수 있다. 만약 [그림 6]의 3가지 유형에서 [그림 5]의 주 부호책 벡터 양자화 결정 알고리듬으로 인해 ▨ 블록이 양자화되지 않는다면 현재 블록이 배경 블록이라는 것을 나타내는 것이므로 이 경우에는 현재 블록 x가 3면이나 4면이 아닌 2면을 이용한 사이드 매치 벡터 양자화를 수행하게 된다. 그러므로 본 논문에서 제안하는 방법으로 벡터 양자화를 실행하면 2면, 3면, 4면(4면 중 2면은 4 화소)을 사용하는 다중 면 사이드 매치 유한상태 벡터 양자화를 수행하게 된다.

5. 상태 부호책의 크기

SMVQ를 사용하는 경우 주 부호책의 부호단어 개수가 256개이라면 상태 부호책은 256×256개가 작성된다. 이 때 각각의 상태 부호책의 부호단어 개수는 주 부호책보다는 작은 크기를 갖게 되는데, 만약 상태 부호책의 부호단어 크기가 16개라면 4비트를 사용하여 부호단어 인덱스를 전송하면 된다. 즉, 상태 부호책을 사용하지 않으면 주 부호책의 부호단어 개수인 256으로 인해 8비트를 전송해야 하지만, 4비트 만을 전송하게 됨으로써 영상 압축의 효과가 있는 것이다.

N_f개의 부호단어를 가지는 상태 부호책 $SC_s = \{y_1, y_2, y_3, \cdots, y_{Nf}\}$에서 부호단어들을 사이드 매치 왜곡 값의 오름차순으로 정렬시켜 놓았을 때, 배경 블록에 대해 y_1이 최종 전송 인덱스로 선택되는 경우가 대다수이다. 이는 배경 블록에 대해서 각 상태 부호책의 부호단어 개수를 윤곽선 블록과 같은 개수로 할 필요가 없다는 것을 의미한다. 그러므로 배경 블록에 대한 각 상태 부호책은 부호단어를 y_1만 사용하기로 한다. 이와 같은 방법으로 각 상태 부호책은 주 부호책으로부터 생성이 되는데, 각 상태 부호책은 단 한 개만의 부호단어를 가지고 있으므로 결국 배경 블록에 대해서는 전송 비트가 없고 단지 어느 블록이 배경 블록인지를 나타내는 윤곽선 맵의 정보만 복호화기(decoder)로 전송되어 그만큼 전송 비트 수를 크게 줄일 수 있다.

6. 가변길이 인덱스 부호화

앞 절에서 제안한 MSMVQ를 사용하여 상태 부호책의 부호 단어 인덱스를 복호화기로 전송할 때, 본 논문에서는 전송 비트 수를 더 줄이기 위해 인덱스 부호화에 고정길이 부호화 대신에 허프만 부호화를 사용한 가변 길이 부호화를 수행하는데 허프만 부호화를 할 때 부호 생성을 쉽게 하고 속도를 높이기 위해 다음과 같은 알고리듬을 사용한다.

6.1 부호 단어 길이(CWL) 표 작성

부호책의 부호 단어 인덱스 집합 I를 정의하면, 집합 I는 $I = \{i_1, i_2, \ldots, i_n\}$이고, 각 인덱스 원소들에 대한 돗수분포 집합은 $P(i_j) = p_j \ \ (j = 1,2,...,n)$이고, 각 돗수분포가 $p_1 \geq p_2 \geq \cdots \geq p_n$라고 하면, 이와 같은 인덱스 집합 I가 주어졌을 때 I에 대한 부호 단어 길이(Code Word Length:CWL) 표의 작성을 위해 다음과 같은 알고리듬을 적용한다.

[단계 1] 기존 허프만 부호화[16] 방식과 같이 돗수 분포에 따라 배열된 허프만 표에서 밑에서부터 두 인덱스를 묶어 나가면서 합산한 돗수 값에 따라 새로운 병합 인덱스 c_j를 원래의 인덱스들 사이에 위치시킨다. [표 1]에

서 첫 열(column)의 i_5와 i_6 두 인덱스를 묶으면 새로운 병합 인덱스 c_1이 만들어지고 이 c_1 인덱스의 돗수 값이 9이므로 두 번째 열의 i_4 밑에 놓이게 된다.

표 1. 허프만 표의 예

인덱스	도수	인덱스	도수	인덱스	도수	인덱스	도수	인덱스	도수
i_1	80	i_1	80	i_1	80	i_1	80	i_1	80
i_2	25	i_2	25	c_2	28	c_3	47	c_4	75
i_3	22	i_3	22	i_2	25	c_2	28		
i_4	19	i_4	19	i_3	22				
i_5	6	c_1	9						
i_6	3								

이와 같이 최종 두 인덱스만 남을 때까지 인덱스들을 줄여 나간다. [표 1]의 작성 과정 중에 [표 2]의 CWL표를 동시에 작성하는데 [표 2]에서 첫 열은 두 인덱스 합병시 돗수가 작은 것, 두 번째 열은 돗수가 큰 것이고, 세 번째 열은 그에 해당하는 부호길이를 나타내는 열로서 차후에 계산한다. 우선 [표 1]의 인덱스들을 합병해 나가면서 [표 2]의 CWL표도 동시에 작성한다. 이때 CWL표에 대해 처리 속도를 높이기 위해 병합 인덱스$(c_j\ ;\ j=1,2,\ldots.)$들에 대한 주소는 현재 j의 반전(inverse)된 값을 가지도록 한다. 예를 들어, 4 비트 워드로서 인덱스들을 나타낸다면 c_1은 0001의 반전값인 1110, c_2는 1101로 주소를 가지게 한다. 이로써 원(source) 인덱스$(i_j\ ;\ j=1,2,\ldots)$와 병합 인덱스와의 MSB(Most Significant Bit)만 검사하면 두 인덱스들을 쉽게 구별해 낼 수 있다.

표 2. 부호 단어 길이 표의 예

행	i_j	i_{j-1}	부호길이
1	i_6	i_5	4
2	c_1	i_4	3
3	i_3	i_2	3
4	c_2	c_3	2
5	c_4	i_1	1

[단계 2] [표 2]의 CWL 표에서 부호길이를 할당하는 알고리듬은 [그림 7]과 같다. CWL 표에서 원래의 인덱스와 병합 인덱스의 구분은 그 인덱스 주소의 MSB만을 검사하면 되므로 [그림 7]과 같은 알고리듬을 적용했을 때의 결과적인 부호길이가 [표 2]의 3열에 나타나 있다. [표 2]에서 제 5행의 경우, 병합 인덱스 c_4의 MSB 비트가 1이므로 쉽게 검색이 가능하고 바로 이 인덱스의 주소 값을 반전(inverse)시킨 값인 4열에 부호 길이 2를 할당한다. 나머지 행의 부호 길이도 [그림 7]의 알고리듬에 따라 값을 할당한다.

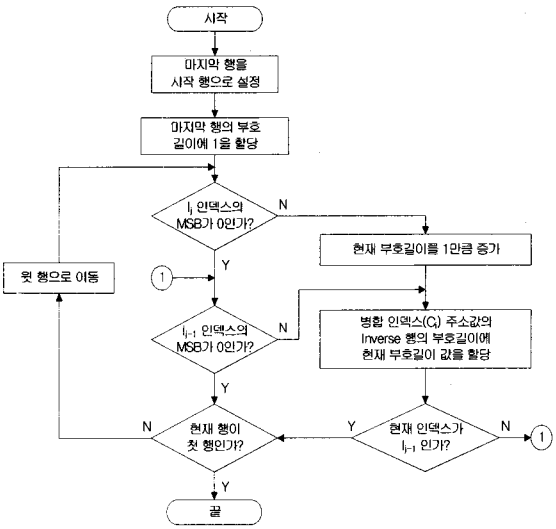


그림 7. 부호 길이 할당 알고리듬

6.2 허프만 트리 부호 생성

기존의 허프만 부호화는 [표 1]과 같은 허프만 표에 따라 인덱스가 두 개만 남을 때까지 병합시키고 다시 병합된 순서의 역으로 부호를 할당하기 때문에 부호 할당 알고리듬의 수행이 쉽지 않고 비교적 시간이 걸리는 편이다. 본 논문에서는 부호 생성을 빠르게 하기 위해 [표 2]에서의 부호길이가 결정되면 그 부호길이에 따라 허프만 부호화를 수행하는데, [그림 8]과 같은 알고리듬을 사용하여 허프만 부호를 생성한다. [그림 8]의 흐름도에 따라서 [표 2]의 원(source) 인덱스$(i_j\ ;\ j=1,2,\ldots)$에 대한 허프만 부호화를 수행한 결과가 [표 3]에 나타나 있다. [표 3]에서 먼저 i_1 인덱스의 부호길이만큼 0을 할당하므로 허프만 부호는 "0"이 되고, 그 값에 1을 더한 값인 "1"을 i_2에 할당한 후, i_2의 부호 길이가 3이므로 "1" 뒤에 두개의 0을 더 추가하여 "100"으로 허프만 부호가 생성된다. 이와 같은 방법으로 나머지 인덱스에 대해서도 허프만 부호를 만든다.

이러한 허프만 트리의 부호생성 알고리듬에서는 가장 짧은 인덱스의 부호는 "00...0"가 되고, 가장 긴 인덱스는 "11...1"의 부호로 만들어지므로 자기 오류 검출(self error checking) 기능을 가지고 있다. 즉 허프만 부호 생성과정의 이상 여부를 알 수 있고 가장 긴 부호 길이의 마지막

원(source) 인덱스 부호를 검사함으로써 허프만 트리의 최종 부호가 생성되었는지를 쉽게 알 수 있다.

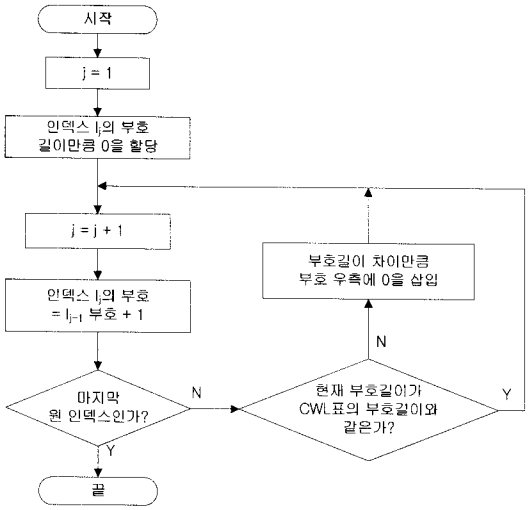


그림 8. 허프만 트리의 부호 생성 알고리듬

표 3. 허프만 부호 생성의 예

부호길이	인덱스	허프만 부호
1	i_1	0
3	i_2	100
3	i_3	101
3	i_4	110
4	i_5	1110
4	i_6	1111

7. 실험 결과

제안 알고리듬의 평가는 512×512 크기의 Zelda, Lenna, Bridge, Peppers 영상에 대해 수행하였다. Zelda와 Lenna 영상은 비교적 윤곽선이 적고, Bridge와 Peppers 영상은 전체적으로 윤곽선 부분이 많은 영상으로, 윤곽선의 양에 의한 시뮬레이션 결과를 비교하기 위해서 각 영상들을 선택하였다. 제안 알고리듬과 기존 알고리듬을 비교하기 위해 SMVQ[8]와 TSMVQ[9]를 실험 영상에 적용하였고, 객관적 평가를 위해 복원 영상에 식 (1)의 PSNR을 계산하였다.

$$PSNR = 10\log_{10}\frac{255^2}{MSE}[\mathrm{dB}] \qquad (1)$$

SMVQ와 TSMVQ, 제안 알고리듬인 MSMVQ에서 모든 주 부호책의 크기는 256으로, 상태 부호책의 크기는 16으로 하였고, 제안 알고리듬의 배경 블록에 대한 상태 부호책의 크기는 1로 하였다.

[표 4]는 기존 알고리듬인 SMVQ와 TSMVQ, 그리고 제안 알고리듬의 4개 영상에 대한 실험 결과를 식 (1)을 이용하여 PSNR 값을 계산한 것이다. 표에서 알 수 있듯이 Zelda 영상은 기존의 유한상태 벡터 양자화 기법인 SMVQ에 비하여 2.47 dB, TSMVQ에 비하여 0.64 dB만큼 화질이 더 좋았으며, Lenna 영상에 대해서는 SMVQ에 비하여 3.28 dB, TSMVQ에 비하여 0.92 dB, Peppers 영상에 대해서는 SMVQ에 비하여 3.52 dB, TSMVQ에 비하여 2.69 dB, Bridge 영상에 대해서는 SMVQ에 비하여 2.36 dB, TSMVQ는 1.16 dB만큼 복원 영상의 화질이 좋았다. 4개의 영상 모두에서 SMVQ와 TSMVQ보다 제안 알고리듬의 비트율이 낮았고, 전송 비트 수가 더 적은 상황에서 결과 PSNR이 제안 알고리듬이 더 좋았다. 이로써 본 논문에서 제안한 알고리듬이 더 좋은 성능을 나타냄을 알 수 있다.

표 4. SMVQ와 TSMVQ, 제안 알고리듬(MSMVQ)의 결과 PSNR(dB) 비교

영상	SMVQ	TSMVQ	제안 알고리듬
Zelda	32.05 (0.25 bpp)	33.88 (0.2 bpp)	34.52 (0.17 bpp)
Lenna	31.69 (0.25 bpp)	34.05 (0.19 bpp)	34.97 (0.16 bpp)
Bridge	28.82 (0.25 bpp)	30.02 (0.21 bpp)	31.18 (0.20 bpp)
Peppers	29.36 (0.25 bpp)	30.19 (0.21 bpp)	32.88 (0.19 bpp)

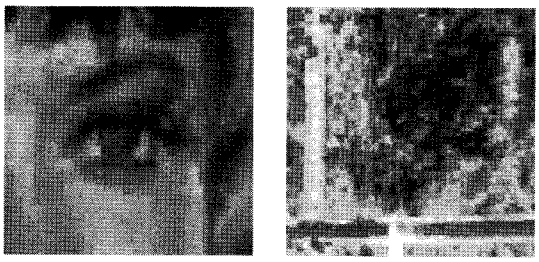
기존 알고리듬과 제안 알고리듬에 대한 처리 시간은 SMVQ의 시뮬레이션 시간을 1로 봤을 때, TSMVQ는 약 1.03, 제안 알고리듬은 약 1.08 정도 더 오래 걸렸다. SMVQ는 기본적인 2면을 이용한 사이드 매치만으로 벡터 양자화를 하지만, TSMVQ는 3면을 이용하여 사이드 매치를 함으로 처리 시간이 약간 길어졌고, 본 논문에서 제안한 알고리듬은 사이드 매치를 수행하기 전에 윤곽선 맵을 작성하고, 다중 면 사이드 매치를 함으로써 TSMVQ보다 더 많은 시간을 요구한다. 하지만 이와 같은 시간 차이는 그리 큰 값이 아니기 때문에 처리 시간과 결과 영상의 화질의 상관관계를 볼 때 처리 시간은 그리 큰 요소가 되지 못한 다는 것을 알 수 있다.

[그림 9]에는 4개의 실험 영상 중 Zelda와 Bridge 영상에 대한 SMVQ와 TSMVQ, 본 논문에서 제안한 알고리듬의 [표 4]에서의 시뮬레이션 결과 영상이 나타나 있다. (a)와 (b)는 원 영상이고, (c)와 (d)는 각 알고리듬간의 비

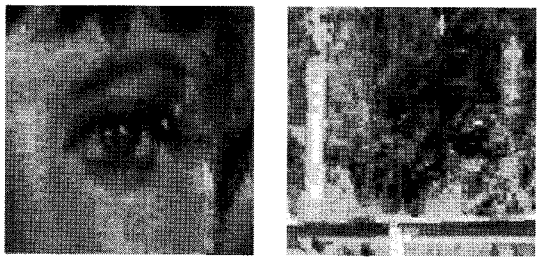
교를 확실히 하기 위해 원영상의 한 부분을 확대한 그림이며, 그 아래는 각 알고리듬에 대한 결과 영상 중 (c)와 (d) 부분을 확대한 그림이다. 그림에서 각각의 SMVQ, TSMVQ 결과 영상과 제안 알고리듬의 결과 영상을 비교해 보면 영상의 윤곽선(edge) 영역에서 다른 알고리듬보다 본 논문의 제안 알고리듬이 더 적게 왜곡이 일어난 것을 볼 수 있다.

(a) (b)

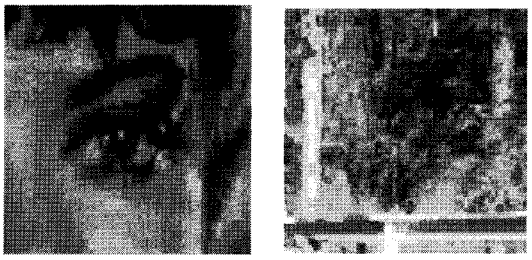
(c) (d)



(e) (f)



(g) (h)



(i) (j)

그림 9. "Zelda"와 "Bridge" 영상에 대한 SMVQ, TSMVQ, 제안 알고리듬의 비교
(a) "Zelda" 원 영상
(b) "Bridge" 원 영상
(c) "Zelda" 확대 영상
(d) "Bridge" 확대 영상
(e),(f) SMVQ의 결과 확대 영상
(g),(h) TSMVQ의 결과 확대 영상
(i),(j) 제안 알고리듬(MSMVQ)의 결과 확대 영상

8. 결 론

본 논문에서는 영상의 벡터 양자화를 위한 새로운 부호화 알고리듬을 제안하였다. 영상에 대해 Sobel 연산자를 사용하여 윤곽선을 구하고, 영상을 4×4 영역으로 분할한 후, 윤곽선 맵을 작성하였다. 각 블록에 대해 DCT의 두 AC 계수를 사용하여 윤곽선 영역에 대해 다시 16개로 분류하고 배경 블록과 윤곽선 블록에 대해 각각 따로 주 부호책과 상태 부호책을 작성하였다. 현재 블록에 대해 2면 또는 3면으로 사이드 매치를 실행하기 위해서 다중 면 사이드 매치 유한상태 벡터 양자화기(MSMVQ)를 제안하였으며, 이를 사용함으로써 증가되는 전송 비트의 수를 줄이기 위해 주 부호책 벡터 양자화 결정 알고리듬을 제안하였고, 각 벡터 양자화 인덱스를 전송할 때 가변 길이 부호화를 사용하였다. 시뮬레이션 결과 본 논문에서 제안한 방법으로 영상 부호화를 수행했을 경우 SMVQ나 TSMVQ 알고리듬과 비교하여 더 좋은 화질의 영상을 얻을 수 있었다.

참고문헌

[1] R. M. Gray, "Vector Quantization," IEEE ASSP Mag., vol. 1, pp. 4-29, April 1984.

[2] J. Makhoul et al., "Vector Quantization in Speech Coding," Proc. IEEE, vol. 73, pp. 1551-1588, Nov.

1985.

[3] N. M. Nasrabadi and R. B. King, "Image Coding Using Vector Quantization : a review," IEEE Trans. Commun., vol. COM-36, pp. 957-971, Aug. 1988.

[4] D. J. Vaisey and A. Gershot, "Variable Block-Size Image Coding," Proc. ICASSP, pp. 1051-1054, 1987.

[5] P. Yu and A. N. Venetsanopoulos, "Hierarchical Multirate Vector Quantization for Image Coding," Signal Processing : Image Commun., vol. 4, no. 6, pp. 497-505, Nov. 1992.

[6] A. Gersho and R. M. Gray, Vector Quantization and Signal Compression, Boston : Kluwer Academic Publishers, 1992.

[7] J. Foster, R. M. Gray, and M. O. Dunham, "Finite State Vector Quantization for Waveform Coding," IEEE Trans. Inform. Theory, vol. IT-31, pp. 348-355, May 1985.

[8] T. Kim, "New Finite State Vector Quantizers for Images," Proc. ICASSP, pp. 1180-1183, 1988.

[9] H. Wei, P. Tsai, and J. Wang, "Three-Sided Side Match Finite-State Vector Quantization," IEEE Trans. On Circuits And Systems For Video Technology, vol. 10, no. 1, Feb. 2000.

[10] Hsuan T. Chang, "Gradient Match and Side Match Fractal Vector Quantizers for Images," IEEE Trans. On Image Processing, vol. 11, no. 1, Jan. 2002.

[11] 임유두, "주변의 움직임 벡터를 사용한 비디오 에러 은닉 기법," 한국통신학회논문지, 28권 3C호, pp. 257-263, 2003. 3.

[12] 이귀상, 유재명, "동일 영역 움직임을 적용한 에러 복원기술 연구," 전자통신기술논문지, 7권 1호, pp. 19-24, 2004. 1.

[13] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," IEEE Trans. Commun., vol. COM-28, pp. 84-95, Jan. 1980.

[14] B. Ramamurthi and A. Gersho, "Classified Vector Quantization of Images," IEEE Trans. Commun., vol. COM-34, no. 11, pp. 1105-1115, Nov. 1986.

[15] C. H. Yim and J. K. Kim, "A Simple DCT-CVQ Based on Two DCT Coefficients," PCS 88, Picture Coding Symp., pp. 8.11-1 through 8.11-2, Torino, Italy, Sept. 1988.

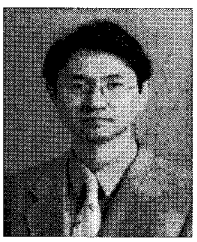
[16] D. A. Huffman, "A Method for The Construction of Minimun Redundancy Codes," Proc. IRE, vol. 40, no. 10, pp. 1098-1101, Sept. 1952.

조 성 환(Seong-Hwan Cho) **[정회원]**

- 1980년 2월 : 성균관대학교 전자공학과 (공학사)
- 1982년 2월 : 성균관대학교 대학원 전자공학과(공학석사)
- 1991년 8월 : 성균관대학교 대학원 전자공학과(공학박사)
- 1982년~1985년 : 해군사관학교 전기 및 전자공학과 전임강사
- 1997년 : 미국 Columbia 대학 CATT Visiting Scholar
- 1985년 ~ 2002년 : 동서울대학 컴퓨터공학과 부교수
- 2003년 ~ 현재 : 금강대학교 교양학부 컴퓨터전공 부교수

<관심분야>
영상처리, 신경회로망, 패턴인식, DRM 등

김 응 성(Eung-Sung Kim) **[정회원]**



- 1989년 2월 : 성균관대학교 전자공학과 (공학사)
- 1992년 2월 : 성균관대학교 대학원 전자공학과(공학석사)
- 1998년 2월 : 성균관대학교 대학원 전자공학과(공학박사)
- 1995년 3월~2000년 2월 : 성균관대학교 과학기술연구소 연구전담요원
- 2000년 3월~현재 : 경기공업대학 컴퓨터정보시스템과 부교수

<관심분야>
영상처리, 영상통신, 신경회로망 등