벡터양자화기의 코드북을 구하는 새로운 고속 학습 알고리듬

A New Fast Training Algorithm for Vector Quantizer Design

이 대 룡*, 백 성 준*, 성 굉 모**
(Daeryong Lee*, Seongjoon Baek*, Koengmo Sung**)

요 약

본 논문에서는 코드북 학습 알고리듬의 대표적인 LBG 알고리듬의 탐색시간을 줄이기 위한 새로운 고속 학습 알고리듬을 제안한다. 제안한 알고리듬은 각 학습데이타가 모든 코드워드를 탐색하지 않고, 먼저 첫 번째 단계에서 각 학습데이타의 주위에 있는 일정한 개수의 코드워드에 대한 인덱스(index) 정보를 저장하고, 다음 단계에서부터는 이 인덱스가 가리키는 코드워드만을 탐색대상으로 함으로써 학습시간을 줄이는 것이다. 제안한 알고리듬을 기존의 고속 탐색 알고리듬인 FSLBG 알고리듬과 비교하면 제안한 알고리듬이 더 짧은 학습시간으로 더 좋은 성능을 갖는 코드북을 얻을수 있음을 보인다. 또한 제안한 알고리듬을 LBG 알고리듬과 비교하면 영상데이타에 대해 코드북의 크기가 256인 경우에는 약 6%, 코드북의 크기가 1024인 경우에는 약 1.6%인 16개의 코드워드만을 탐색대상으로 해서 PSNR(peak signal-to-noise ratio)면에서 거의 성능이 같은 코드북을 생성할수 있음을 보이고 있다.

ABSTRACT

In this paper we propose a new fast codebook training algorithm for reducing the searching time of LBG algorithm. For each training data, the proposed algorithm stores the indexes of codewords that are close to that training data in the first iteration. It reduces computation time by searching only those codewords, the indexes of which are stored for each training data. Compared to one of the previous fast training algorithm, FSLBG, it obtains a better codebook with less execution time. In our experiment, the performance of the codebook generated by the proposed algorithm in terms of peak signal-to-noise ratio(PSNR) is very close to that of LBG algorithm. However, the codewords to be searched for each training data of the proposed algorithm is only about 6%, for a codebook size of 256 and 1.6%, for a codebook size of 1024, of LBG algorithm.

I. 서 론

벡터양자화 기법은 음성압축이나 영상압축, 음성 그식, 화자인식, 패턴인식등에 가장 널리 쓰이고 쓰이는 알고리듬의 하나이다[1]. 벡터양자화의 가장 큰 장점 중의 하나는 이에 쓰이는 복호화기와 부호화기의 구조가 매우 간단하다는 것이다. 부호화하고자 하는 음성데이타나 화상데이타는 먼저 벡터열의 집합으로 구성된다. 그런 다음에 이 입력된 벡터열을 이용하여 반복적인 클러스터링 알고리듬인 LBG 알고리듬 등의 알고리듬으로 코드북을 구성하게 된다. 이 코드북은 미리 정해진 개수의 코드워드를 그 원소로 구성된다. 주어진 입력 벡터는 각각 이들 중의 가장 거리가 가까운 하나의 코드워드로 양자화된다. 데이타를 전송할 때는 원래의 음성데이타나 화상데이타가 아니라 이들 코드워드의 이진 표현값을 전송함으로써 데이타압축이 이루어지는 것이다. 복호화는 전송된 코드워드의 이진값을 보고서 이에 해당하는 코드워드의 값으로 치환함으로써 이루어진다. 이러한 코드워드의 이진값은 단지 코드북 내에 있는 코드워드의 주소를 가리키는 것으로 생각할 수 있다.

벡터양자화 기법의 핵심적인 사항은 주어진 학습데이타를 이용해서 어떻게 하면 가장 좋은 코드북을 생성할 수 있는 가이다. 코드북을 생성하는 알고리듬(학습알고리듬) 중에서 지금까지 가장 널리 쓰이는 알고리듬은 k 평균 알고리듬 또는 Linde, Buzo, Gray가 제안한 알고리듬으로 k 평균 알고리듬과 본질적으로 동일한 LBG 알고리듬이다[2, 3]. 이 알고리듬은 학습데이타와 이에 대응하는 코드워드 사이의 오차를 반복적으로 감소시켜나가는 알고리듬이다.

LBG 알고리듬의 가장 큰 단점은 학습의 각 단계(iter-

*서울대학교 전자공학과 박사과정
**서울대학교 교수

접수일자:1996년 7월 20일

ation)에서 각각의 학습데이타에 대해 가장 가까운 코드워드를 찾기 위하여 코드북의 모든 코드워드와의 사이에 대한 거리오차를 계산해야 한다는 것이다. 부호화 단계에서 이 문제를 해결하기 위하여 많은 고속 부호화 알고리듬이 제안되어 있다[4-7]. 그러나 많은 고속 부호화 알고리듬은 주어진 코드북에 대한 부호화 속도를 증가시키기 위하여 여러 가지 전처리를 하게되므로 이 전처리에 걸리는 시간에 의해 학습알고리듬에 고속부호화 알고리듬을 이용하게 되면 학습속도가 많이 느려지게 된다. 그러나 C.-D. Bei와 R. M. Gray가 제안한 PDE(partial distortion elimination) 알고리듬[4]은 전처리 과정이 필요하지 않으므로 학습알고리듬에도 그대로 이용될수 있다. 코드북을 만드는 학습단계에서 속도를 빠르게 하기 위한 고속 학습 알고리듬은 그렇게 많이 제안되어 있지 않다. 그러한 고속 학습 알고리듬의 대표적인 것의 하나는 R.-F. Chang, W.-T. Chen과 J.-S. Wang이 제안한 FSLBG 알고리듬이 있다[8]. 이 알고리듬은 메모리가 있는 벡터양자화기의 하나인 FSVQ 알고리듬의 상태 개념을 코드북 학습알고리듬에 도입한 것이다. 현재 단계에서 각각의 학습데이타와 가장 거리오차가 작은 코드워드를 찾고자 할 때 이전 단계에서 구해진 가장 가까운 코드워드 가까이에 있는 일정한 개수만큼의 코드워드만을 대상으로 해서 찾는 방법이다. 이 FSLBG 알고리듬은 필요한 계산량을 많이 줄일수 있지만 그 구해진 코드북의 성능이 기존의 LBG 알고리듬에 비해 많이 떨어지는 단점이 있다. 또한 FSLBG 알고리듬은 매 단계(iteration) 마다 코드워드들 사이의 거리오차를 구해 이것을 소팅하는 여분의 오버헤드를 필요로 한다.

본 논문에서는 앞에서 언급한 2가지 알고리듬의 단점을 개선한 새로운 학습알고리듬을 제안한다. 제안한 알고리듬도 역시 FSLBG 알고리듬과 같이 각 단계에서 각각의 학습데이타에 대해 모든 코드워드를 탐색하는 것이 아니라 제한된 개수만큼의 코드워드만 탐색하게 된다. 그러나 FSLBG 알고리듬과 같이 이전 단계에서 구해진 해당 코드워드 주위의 일정한 개수의 코드워드를 탐색하는 것이 아니라 먼저 첫 번째 단계에서 각 학습데이타와 가장 가까운 일정한 개수의 코드워드의 index에 대한 정보를 저장하고 있다가 다음 단계부터는 그 학습데이타에 대해서는 계속해서 이 일정한 개수의 index에 해당하는 코드워드만을 대상으로 탐색을 수행하는 것이다. 이렇게 함으로써 각 학습데이타에 대해 FSLBG 알고리듬과 비교해 보다 더 정확한 코드워드를 찾을수 있는 것이다. 이러한 방법으로 제안한 알고리듬은 FSLBG 알고리듬에 비해 더 적은 계산량으로 학습데이타에 대한 거리오차가 더 작은 코드북을 생성할수 있다. 또한 LBG 알고리듬에 비해서도 그 성능차가 거의 없다는 것을 알수 있다.

Ⅱ. 벡터양자화를 위한 LBG 알고리듬과 FSLBG 알고리듬

벡터양자화라는 것은 k차원의 유클리드 공간 R^k에서 어떤 유한한 집합 $Y=\{y_i, i=1, 2, \cdots, N\}$에 대한 사상으로 정의할수 있다. 이 유한 집합 Y를 코드북이라 부른다. 코드북의 각 원소 $y_i=(y_0, \cdots, y_{(k-1)})$를 코드워드라 부른다. 각 코드워드는 k차원 유클리드 공간의 일정한 영역을 대표하게 된다. LBG 알고리듬과 FSLBG 알고리듬은 이러한 코드북을 구하는 알고리듬들이다. 전송로상에서 벡터양자화기를 구현할 때는 크게 2가지 과정이 필요하게 된다. 하나는 부호화기이고 다른 하나는 복호화기이다. 이들 각각에는 학습알고리듬으로 구한 동일한 코드북이 존재한다. 부호화기는 각각의 입력 벡터 $x=(x_0, \cdots, x_{(k-1)}) \in R^k$에 대해 이와 가장 거리오차가 작은 코드워드 $\hat{x}=y_i$에 대한 index i를 할당한다. 복호화기에서는 이 index를 사용해 이에 해당하는 코드워드 $\hat{x}$를 찾게된다. 전송 비트율 조정은 코드북에 대한 크기를 조정함으로써 이루어진다. 입력 벡터 x와 이에 해당하는 코드워드 $\hat{x}$사이의 거리오차 $d(x, \hat{x})$는 일반적으로 다음과 같은 제곱거리오차

$$d(x, \hat{x}) = \sum_{j=0}^{k-1} (x_j - \hat{x}_j)^2 \tag{1}$$

가 많이 사용된다. 그러나 LPC 계수 등을 입력 벡터로 사용하는 음성데이타 압축과 같은 경우에는 여러 가지 다른 거리오차가 사용된다.

LBG 알고리듬은 주어진 초기 코드북을 반복적으로 개선시켜서 최종적인 코드북을 얻는 알고리듬이다. 이것은 국소 최적 코드북으로 수렴하는 것을 보장한다. 그러나 그 최종 코드북의 성능은 주어진 초기 코드북에 의해 많은 영향을 받는다. 그러므로 랜덤 초기화, 최대거리 초기화, PNN, splitting 등의 많은 초기화 알고리듬이 있는데 이들 중에서 splitting 알고리듬이 가장 널리 쓰이고 있다. 그러나 최근에 제안된 최대거리 초기화 알고리듬은 코드북의 크기가 256 이상인 경우에는 splitting 알고리듬보다 더욱 뛰어난 성능을 보이는 걸로 알려져있다[9]. LBG 알고리듬은 크게 2가지 과정으로 이루어진다. 하나는 부호화과정으로 각 학습데이타에 대해 가장 가까운 코드워드를 구해, 이 코드워드에 해당하는 클래스에 이 학습데이타를 등록시키는 과정이고, 다른 하나는 코드워드 갱신 과정으로 각 클래스에 등록된 학습데이타의 중심을 구해 이것을 새로운 코드워드로 치환하는 과정이다.

코드워드 y에 대한 해당되는 클래스를 $S(y)$라고 정의하면 어떤 학습데이타가 코드북의 모든 코드워드 중에서 코드워드 y와의 거리오차가 가장 작다면 이 학습데이타는 클래스 $S(y)$의 원소가 된다. 클래스 $S(y)$의 중심 $cent(S(y))$는 다음과 같이 정의된다.

$$cent(S(y)) = \frac{1}{|S(y)|} \sum_{x \in S(y)} x \tag{2}$$

여기서 $|S(y)|$는 $S(y)$ 클래스에 속하는 학습데이타의 개수를 나타내는 것이다. 모든 학습데이타와 각각의 해당

하는 코드워드들 사이의 평균적인 거리오차 D는 다음과 같이 정의된다.

$$D=\frac{1}{T}\sum_{x} d(x,\hat{x}) \tag{3}$$

여기서 T는 총 학습데이타의 개수이고, $\hat{x}$는 학습데이타 x와 가장 가까운 코드워드이다. 이러한 조건아래에서 LBG 알고리듬은 다음과 같이 구성된다.

step 0: 주어진 초기 코드북을 C_0라 하자. $i=0$, $D_{-1}=\infty$로 둔다.

step 1: 각각의 학습데이타 x에 대해 코드북 C_i의 모든 코드워드를 탐색해 가장 가까운 코드워드 y를 찾는다. 이 학습데이타를 클래스 $S(y)$에 등록한다.

step 2: 전체의 평균 거리오차 D_i를 계산한다. 만약 $(D_{i-1}-D_i)/D_i \le \varepsilon$이면 알고리듬을 끝낸다. 그렇지 않으면 step 3으로 간다.

step 3: 코드북 C_i의 각 코드워드 y^i를 $y^{i+1}=cent(S(y^i))$로 갱신한다. 이 y^{i+1}를 코드북 C_{i+1}에 넣는다.

step 4: i를 $i+1$로 증가시키고 step 1로 간다.

이 중에서 step 1이 부호화과정이고 step 3이 코드북 갱신과정이다. LBG 알고리듬은 step 1에서 보이는 바와 같이 각각의 학습데이타에 대해서 코드워드 개수 N 만큼의 거리오차를 계산해야 하므로 많은 계산량을 필요로 하게 된다.

FSLBG 알고리듬은 LBG 알고리듬의 이러한 많은 계산량을 줄이고자 하는 목적으로 제안되었다. 이것은 LBG 알고리듬의 부호화과정에 필요한 코드워드 탐색범위를 줄이고자 FSVQ 알고리듬의 개념을 이용한 것이다. 만약 i번째 단계의 코드북을 C_i라 하고, 어떤 학습데이타 x에 대한 해당하는 코드워드가 $\hat{x}^i$라 하면, 이 $\hat{x}^i$를 찾기 위해서 C_i의 모든 코드워드를 탐색하는 것이 아니고 C_i의 코드워드 중에서 이전 단계에서 구한 $\hat{x}^{i-1}$와 가장 가까운 N_f개의 코드워드 만을 대상으로 탐색하는 것이다. 최초의 단계에서는 LBG 알고리듬과 동일하다. 즉, 각각의 학습데이타에 대해서 초기 코드북의 모든 코드워드를 탐색한다. 이 다음 단계에서부터는 이전 단계에서 구한 코드워드에 대한 정보를 이용하여 탐색범위를 줄이는 것이다. FSLBG 알고리듬의 기본적인 과정은 다음과 같다.

step 0: 초기 코드북 C_0을 얻는다.

step 1: 각각의 학습데이타 x에 대해 코드북 C_0의 모든 코드워드를 탐색해 가장 가까운 코드워드 $\hat{x}^i$를 찾는다. 평균 거리오차 D_0를 계산한다.

step 2: 코드북 C_0의 각 코드워드 y^0에 대해 새로운 코드워드 $y^1=cent(S(y^0))$를 구하고, 이것을 코드북 C_1에 넣는다. i를 1이라 한다.

step 3: 코드북 C_{i-1}의 각 코드워드 y^{i-1}에 가장 가까운 코드워드 N_f개를 코드북 C_i에 찾아 이것을 $SC(y^{i-1})$이라 한다.

step 4: 각각의 학습데이타 x에 대한 코드워드 $\hat{x}^i$를 $SC(x^{i-1})$내에서만 찾는다.

step 5: 전체 학습데이타에 대한 평균 거리오차 D_i를 계산한다. 만약 $(D_{i-1}-D_i)/D_i \le \varepsilon$이면 알고리듬을 끝낸다. 그렇지 않으면 step 6으로 간다.

step 6: 코드북 C_i의 각 코드워드 y^i를 $y^{i+1}=cent(S(y^i))$로 갱신한다. 이 y^{i+1}를 코드북 C_{i+1}에 넣는다.

step 7: i를 $i+1$로 증가시키고 step 3으로 간다.

FSLBG 알고리듬은 첫 번째 단계를 제외하고는 부호화과정에서 각 학습데이타에 대해 N_f번의 거리오차 계산을 하게된다. 이것은 LBG 알고리듬의 N번에 비해 일반적으로 매우 작은 값이다. 그러나 FSLBG 알고리듬은 이전 단계에서 구한 코드워드와 가장 가까운 N_f개의 현재 단계에 대한 코드워드를 찾기 위한 N^2개의 여분의 거리오차 계산이 필요하게 된다 또한 학습데이타수 T만큼의 기억용량이 이전 단계에서의 코드워드 index를 기억하기 위해 필요하게 된다.

Ⅲ. 제안된 고속 학습 알고리듬

제안된 알고리듬 역시 LBG 알고리듬의 학습 속도를 빠르게 하고자 하는 것이다. FSLBG 알고리듬은 LBG 알고리듬의 계산량을 감소시키기 위하여 하나의 가정을 도입했다. 그것은 현재 단계(iteration)에서의 어떤 학습데이타와 가장 가까운 코드워드는 이전 단계에서 구한 코드워드의 주위에 있다는 것이었다. 이것은 메모리가 있는 벡터양자화기법인 FSVQ의 개념을 도입한 것이다. 제안한 알고리듬에서는 2가지 가정을 근거로 한다. 하나는 어떤 데이타에 가장 거리오차가 작은 코드워드는 학습 단계가 진행되어도 최초의 단계에서 이 데이타와 가까운 영역의 어떤 코드워드 중의 하나라는 것이다. 다른 하나는 각각의 코드워드가 대표하고 있는 k차원의 공간영역은 학습단계가 진행되어도 별로 바뀌지 않을 것이라는 것이다. 즉, 각 코드워드의 위치는 학습단계가 진행되어도 그들이 대표하는 영역을 벗어나지 않을 것이라는 것이다.

이러한 가정을 바탕으로 제안한 알고리듬은 먼저 첫 번째 단계에서는 각 학습데이타에 대해 모든 코드워드를 탐색하여 가장 거리오차가 작은 코드워드 N_f개를 구한다. 이렇게 구해진 코드워드들에 대한 index를 메모리에 저장해 둔다. 그런 다음 두 번째 학습단계부터는 모든 코드워드를 탐색하지 않고 메모리에 저장된 index에 해당하는 코드워드만을 탐색함으로써 계산량을 줄이는 것이다. 각 학습단계가 진행하면 당연히 각 코드워드의 값들은 변하게 될 것이다. 그러나 그 변화는 어떤 2개의 코드워드의 위치를 서로 바꿀 정도로 변할수는 없다는 가정

에 근거를 둔 것이다. 즉, 각 코드워드가 대표하는 영역이 학습단계가 진행함에 따라 이전단계에서의 인접 코드워드의 대표 영역을 어느 정도 침범할 뿐이지 그 인접영역을 넘을 수는 없다는 것이다. 그러므로 제안한 알고리듬의 구성은 다음과 같다.

step 0: 초기 코드북 C_0을 얻는다.

step 1: 각각의 학습데이타 x에 대해 코드북 C_0의 모든 코드워드를 탐색해 가장 가까운 코드워드 N_f개를 찾는다. 이 코드워드에 대한 index의 집합을 $SC(x)$라 한다. 평균 거리오차 D_0를 계산한다.

step 2: 코드북 C_0의 각 코드워드 y^0에 대해 새로운 코드워드 $y^1=cent(S(y^0))$를 구하고, 이것을 코드북 C_1에 넣는다. i를 1이라 한다.

step 3: 각각의 학습데이타 x에 대한 코드워드를 $\hat{x}^i$를 $SC(x)$에 들어있는 N_f개의 index에 해당하는 코드워드 중에서 찾는다.

step 4: 전체 학습데이타에 대한 평균 거리오차 D_i를 계산한다. 만약 $(D_{i-1}-D_i)/D_i \le \varepsilon$이면 알고리듬을 끝낸다.

step 5: 코드북 C_i의 각 코드워드 y^i를 $y^{i+1}=cent(S(y^i))$로 갱신한다. 이 y^{i+1}를 코드북 C_{i+1}에 넣는다.

step 6: i를 $i+1$로 증가시키고 step 3으로 간다.

총 학습데이타의 개수가 T라고 하고, 코드북의 크기가 N이라 하면, LBG 알고리듬은 각 단계에서 TN개의 거리오차계산이 필요하지만 제안한 알고리듬은 두 번째 단계에서부터는 단지 N_f개의 거리오차계산이 필요하게된다. 물론 각 학습데이타에 대한 해당 코드워드들의 index를 저장하기 위한 TN_f개의 여분의 기억용량이 필요하게 된다. FSLBG 알고리듬과 비교하면 제안한 알고리듬은 코드워드에 대한 소팅작업이 필요 없게 되어 각 단계에서 N^2개의 거리오차 계산량과 소팅시간이 절약됨을 알 수 있다. 즉, FSLBG 알고리듬은 매 단계마다 오버헤드가 필요함에 비해 제안된 알고리듬은 첫 번째 단계에서의 오버헤드만이 필요하다. 메모리 용량의 입장에서 보면 FSLBG 알고리듬은 각 학습데이타에 대한 해당 코드워드의 index를 저장하기 위한 T개의 기억용량과 이전 단계의 각 코드워드에 가장 가까운 N_f개의 코드워드 index를 저장하기 위한 NN_f 개의 기억용량이 필요하므로 일반적으로 제안한 알고리듬이 더욱 많은 기억용량을 필요로 하게됨을 알수 있다. 코드북의 성능 면에서 보면 제안한 알고리듬이 일반적으로 더 나은 결과를 보임을 다음 장의 실험을 통해 알수 있다. 제안한 알고리듬이 FSLBG 알고리듬에 비해 더 나은 성능을 보일 수 있는 이유의 하나는 어떤 학습데이타에 대한 코드워드 탐색 범위는 FSLBG 알고리듬이 이전 단계에서 구한 코드워드에 가까운 코드워드를 대상으로 함에 비해 제안한 알고리듬은 그 탐색범위를 그 학습데이타 주위의 코드워드를 대상으로 함으로써 더욱 정확한 코드워드를 찾을 수 있다는 데에 있다. 또한 제안한 알고리듬은 모든 종류의 거리오차에 대해 동일한 계산량 감소의 효과를 얻을 수 있다. 이것은 FSLBG 알고리듬도 마찬가지다. 그러나 PDE 등과 같은 많은 고속부호화 알고리듬은 제곱 평균 거리오차 등의 일정한 종류의 거리오차에 대해서만 적용될수 있다.

Ⅳ. 실험결과

제안한 새로운 학습알고리듬을 기존의 LBG 알고리듬, FSLBG 알고리듬과 비교하기 위하여 3개의 영상데이타를 대상으로 코드북을 생성시켰다. 이 영상은 256의 gray 척도를 갖는 512×512 크기의 흑백 영상인 Lena, baboon 그리고 pepper이다. 모든 실험은 SUN SPARC10 워크스테이션에서 동일한 조건아래에서 수행하였다. 이 코드북으로 부호화 시킨 영상의 성능은 다음과 같이 정의된 PSNR(peak signal to noise ratio)로

$$PSNR = 10\log_{10}\frac{255^2}{MSE}\ \text{dB} \tag{4}$$

측정하였다. PSNR은 영상에 대한 부호화 성능에 가장 널리 쓰이는 성능평가 척도이다. 여기서 MSE는

$$MSE = \left(\frac{1}{512}\right)^2 \sum_{i=0}^{511}\sum_{j=0}^{511}(x_{ij}-\hat{x}_{ij})^2 \tag{5}$$

으로 정의되며, x_{ij}는 원래 영상의 그레이 레벨이고, $\hat{x}_{ij}$는 양자화된 영상의 그레이 레벨이다. 모든 실험에서 문턱값 ε는 0.0001로 고정시켰다. FSLBG 알고리듬과 제안한 알고리듬은 각 학습데이타가 탐색하고자 하는 코드워드의 개수를 줄이는 것이므로 일반적으로 탐색대상 코드워드의 개수가 많아지면 부호화 시킨 영상의 성능은 더욱 좋고, 반면에 학습시간은 더 걸린다. 여기서는 탐색대상 코드워드의 개수 N_f가 8, 16, 24인 경우의 3가지에 대해서 실험하였다. 모든 알고리듬에 대한 실험은 크기가 256과 1024인 2가지의 코드북에 대해 이루어졌다. 또한 앞에서 말한 3가지 알고리듬에는 모두 PDE을 적용할 수 있으므로, 이 개념을 적용한 경우와 그렇지 않은 경우에 대한 학습시간의 차이에 대해서도 실험해 보았다.

표 1과 표 2에 코드북의 크기가 256인 경우의 PDE을 사용하지 않은 경우의 제안한 알고리듬과 기존의 2가지 알고리듬의 PSNR과 학습시간을 비교했다. 표 3은 PDE을 사용한 경우의 학습시간을 비교한 것이다. PDE을 사용한 경우와 그렇지 않은 경우의 PSNR은 동일하다. 마찬가지로 표 4, 5, 6은 코드북의 크기가 1024인 경우에 3가지 알고리듬을 비교한 것이다. 표 1과 표 4에 보는 것과 같이 같은 코드북 크기에 대해 탐색대상 코드워드의 개수가 많을수록 FSLBG 알고리듬과 제안알고리듬에 의한 코드북의 성능이 점점 좋아짐을 알수 있다. 그러나 제안한 알고리듬으로 구한 코드북의 성능이 18번중 한 번을 제외하고는 모두 더 뛰어남을 알수 있다. FSLBG 알고리

표 1. 코드북의 크기가 256이고 PDE을 사용하지 않은 경우의 PSNR 비교.

	FSLBG			제안 알고리듬			LBG
N_f	8	16	24	8	16	24	
Lena	31.491	31.646	31.683	31.537	31.600	31.660	31.674
baboon	23.908	24.053	24.136	24.069	24.140	24.156	24.174
pepper	31.138	31.294	31.331	31.229	31.310	31.327	31.327

표 2. 코드북의 크기가 256이고 PDE을 사용하지 않은 경우의 학습시간(sec) 비교.

	FSLBG			제안 알고리듬			LBG
N_f	8	16	24	8	16	24	
Lena	93.1	125.1	159.0	58.1	95.3	133.9	783.5
baboon	68.4	101.4	151.9	56.3	91.3	117.3	761.6
pepper	75.3	103.5	132.7	68.2	92.1	110.1	547.3

표 3. 코드북의 크기가 256이고 PDE을 사용한 경우의 학습시간(sec) 비교.

	FSLBG			제안 알고리듬			LBG
N_f	8	16	24	8	16	24	
Lena	54.4	70.8	83.2	44.5	71.4	94.5	324.6
baboon	48.5	70.3	102.1	53.0	83.3	103.2	499.0
pepper	41.4	55.4	68.9	53.8	68.9	80.6	235.8

표 4. 코드북의 크기가 1024이고 PDE을 사용하지 않은 경우의 PSNR 비교.

	FSLBG			제안 알고리듬			LBG
N_f	8	16	24	8	16	24	
Lena	34.340	34.442	34.465	34.418	34.475	34.480	34.490
baboon	25.482	25.632	25.696	25.683	25.746	25.764	25.783
pepper	33.967	34.054	34.051	34.016	34.059	34.064	34.065

표 5. 코드북의 크기가 1024이고 PDE을 사용하지 않은 경우의 학습시간(sec) 비교.

	FSLBG			제안 알고리듬			LBG
N_f	8	16	24	8	16	24	
Lena	449.4	439.7	564.4	198.5	231.9	282.7	1993.1
baboon	445.1	580.4	685.7	181.4	247.1	292.7	2464.7
pepper	436.2	544.4	602.6	179.5	240.8	302.7	2388.7

표 6. 코드북의 크기가 1024이고 PDE을 사용한 경우의 학습시간(sec) 비교.

	FSLBG			제안 알고리듬			LBG
N_f	8	16	24	8	16	24	
Lena	243.7	272.3	373.3	113.1	165.7	213.6	757.5
baboon	289.8	423.8	538.5	134.7	192.9	249.4	1320.8
pepper	231.4	345.9	395.9	110.2	172.1	225.8	910.8

듬은 LBG 알고리듬에 비해 baboon과 같이 변화가 심한 영상에 대해서는 특히 성능 감소가 커지지만 제안한 알고리듬은 이와 같은 현상은 일어나지 않는다. 이것은 FSLBG 알고리듬의 탐색대상 코드워드가 제안한 알고리듬의 대상 코드워드보다 더욱 부정확하다는 것을 의미한다. 또한 제안한 알고리듬으로 구한 코드북의 성능은 탐색대상 코드워드의 개수가 16이상이면 LBG 알고리듬으로 구한 코드북과 거의 차이가 나지 않음을 알수 있다. 표 2와 표 5에서는 PDE(partial distortion elimination)[4]을 사용하지 않은 경우의 코드북을 학습시키는 데 필요한 실행시간을 비교한 것으로 제안한 알고리듬의 학습시간이 가장 작다는 것을 보이고 있다. 제안한 알고리듬과 FSLBG 알고리듬을 비교하면 코드북의 크기가 커질수록 제안한 알고리듬의 학습시간이 FSLBG 알고리듬에 비해 더욱 짧아진다는 것을 알수 있다. 이것은 FSLBG 알고리듬은 매 단계(iteration)마다 코드워드들 사이의 거리오차를 구해 이것을 소팅하는 데 드는 시간이 코드북의 크기가 커지면 점점 증가하기 때문이다. 표 3과 6에서는 PDE을 사용한 경우의 학습시간을 비교한 것으로 코드북의 크기가 256인 경우에는 FSLBG 알고리듬이 제안한 알고리듬에 비해 학습시간이 약간 적지만 코드북의 크기가 1024가 되면 FSLBG 알고리듬의 학습시간이 제안한 알고리듬의 2배 가까이 됨을 알수 있다. 전체적으로 보면 제안한 알고리듬으로 구한 코드북의 성능이 FSLBG 알고리듬에 의한 코드북의 성능보다 더욱 뛰어남을 알수 있으며, 학습시간 또한 제안한 알고리듬의 경우가 더욱 짧다는 것을 알수 있다. 제안한 알고리듬은 성능 면에서나 속도면에서 코드북의 크기가 커지면 더욱 효과적임을 알수 있다. 또한 LBG 알고리듬과 비교하면 제안한 알고리듬은 N_f가 16이고 코드북의 크기가 256인 경우는 PDE을 사용하지 않은 경우는 약 1/8, PDE을 사용한 경우는 약 1/5의 학습시간으로 LBG알고리듬과 거의 비슷한 성능을 보이고, 코드북의 크기가 1024인 경우는 PDE을 사용하지 않은 경우는 약 1/10, PDE을 사용한 경우는 약 1/6의 학습시간으로 LBG 알고리듬과 비슷한 성능을 보임을 알수 있다.

V. 결 론

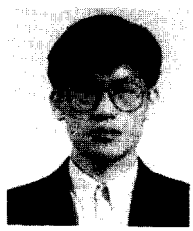
본 논문에서는 음성데이타나 영상데이타의 압축에 널리 쓰이는 LBG 알고리듬의 계산 속도를 빠르게 하기 위한 새로운 고속 학습 알고리듬을 제안하였다. 제안된 알고리듬의 기존의 고속 학습 알고리듬인 FSLBG 알고리듬에 비해 거리오차가 더 적은 코드북을 더욱 적은 계산량으로 구할 수 있음을 보였다. 기존의 고속 알고리듬인 FSLBG 알고리듬은 각 학습데이타에 대해 이전 단계에서 구한 코드워드 가까이에 있는 일정한 개수의 새로운 코드워드만을 탐색대상으로 함으로써 계산량을 감소시킴에 비해, 제안한 알고리듬은 첫 번째 단계에서 각 학습데이타에 가장 가까운 일정한 개수의 코드워드에 대한 index를 저장하고 있다가 다음 단계에서부터는 이들 index가 가리키는 코드워드에 대해서만 탐색을 수행함으로써 각 단계에 필요한 오버헤드를 없앨수 있었으며, 더욱 거리오차가 적은 코드북을 생성할수 있었다. 제안한 알고리듬을 LBG 알고리듬과 비교하면 코드북의 크기가 1024인 경우에 약 10%의 학습시간으로 거의 성능이 같은 코드북을 구할수 있었다. 이 경우 LBG 알고리듬은 각 학습데이타에 대해 1024번의 거리오차 계산을 필요로 함에 비해 제안한 알고리듬은 두 번째 단계부터는 그 1.6% 정도인 단지 16번의 거리오차 계산만으로 거의 비슷한 성능을 갖는 코드북을 얻을수 있다. 제안한 알고리듬에 PDE 알고리듬을 적용하면 더욱 더 계산량을 높일수 있음을 보였다.

참 고 문 헌

1. A. Gersho and R. M. Gray, *Vector Quantization and Signal*

Compression, Boston : Kluwer, 1992.
2. M. R. Anderberg, *Cluster Analysis for Applications*, Academic Press, pp. 165-167, 1973.
3. Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantizer design," IEEE Tran. Comm., COM-28, pp. 84-95, 1980.
4. C.-M. Bei and R. M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," IEEE Trans. Comm., COM-33, pp. 1132-1133, 1985.
5. M. R. Soleymani and S. D. Morgera, "An efficient nearest neighbor search method," IEEE Trans. Comm., COM-35, pp. 677-679, 1987.
6. C.-M. Huang, Q. Bi, G. S. Stiles and R. W. Harris, "Fast full search equivalent encoding algorithms for image compression using vector quantization," IEEE Trans. Image Proc., pp. 413-416, 1992.
7. L. Torres and J. Huguet, "An improvement on codebook search for vector quantization," IEEE Trans. Comm., COM-42, pp. 208-210, 1994.
8. R.-F. Chang, W.-T. Chen and J.-S. Wang, "A fast finite state algorithm for vector quantizer design," IEEE Trans. Signal Proc., SP-50, pp. 221-225, 1992.
9. Ioannis Katsavounidis, C.-C. Jay Kuo and Z., "A new initialization technique for generalized Lloyd iteration," IEEE Signal Processing Letters, Vol. 1, pp. 144-146, Oct. 1994.

▲이 대 룡(Daeryong Lee) 1967년 2월 20일생



1990년 2월 : 한양대학교 전자공학과 졸업(공학사)
1992년 2월 : 서울대학교 전자공학과 석사과정 졸업(공학석사)
1992년 3월~현재 : 서울대학교 전자공학과 박사과정

▲백 성 준(Sungjoon Baek) 1967년 1월 일생

1989년 2월 : 서울대학교 전자공학과 졸업(공학사)
1992년 2월 : 서울대학교 전자공학과 석사과정 졸업(공학석사)
1992년 3월~현재 : 서울대학교 전자공학과 박사과정

▲성 굉 모(Koengmo Sung)
현재 : 서울대학교 교수