論文 90-27-1-14

畫像 벡터 量子化의 코드북 構成을 위한 高速 알고리듬

(Fast Algorithms to Generate the Codebook for Vector Quantization in Image Coding)

李 周 熙*, 丁 海 默**, 李 忠 雄**

(Joo Hee Lee, Hae Mook Jung, and Choong Woong Lee)

要 約

本 論文에서는 벡터 量子化(vector quantization)의 코드북 구성시 所要 시간을 줄이기 위한 알고리듬을 제안한다. 또한 초기 코드북을 정하는데 效率的인 二進 分理(binary splitting)방법을 제안한다. 초기 코드북을 二進 分理 방법으로 구성하고, 이 결과 얻을 수 있는 몇가지 정보와 反復的 最適化(iterative optimization)알고리듬을 이용하여 코드북 구성시간을 대폭 감소시켰다. 제안된 알고리듬은 探索區間(searching range)을 변수로 하여 요구되는 성능에 따라 소요시간을 조정할 수 있다. 제안된 알고리듬으로 코드북을 構成할 때, LBG 알고리듬에 비해 성능이 떨어지지 않으면서 速度를 60배 이상 빠르게 할 수 있었다.

Abstract

In this paper, fast algorithms to generate the codebook of vector quantization in image coding, are proposed. And an efficient algorithm to guess a initial codebook, namely, binary splitting method, is proposed. We generated the initial codebook by binary splitting method and then reduced the searching time using Iterative Optimization algorithm as an alternate to the generalized Lloyd algorithm and several information from binary splitting method. And the searching time and performance can be traded off by varying the searching range. With this proposed algorithm, the computation time can be reduced by a factor of 60 Without any degradation of image quality.

I. 서 론

벡터 量子化(vector quantization)[1,2]는 多次元 벡터(multi-dimensional vector)들을 量子化 雜音(quantizing error)이 최소가 되도록하는 유한개의 코드벡터들로 대표하는 양자화 방식이다. 벡터 양자화는 다른 符號化 方式보다 帶域 壓縮 效果가 크며, 畫像신호에 벡터 양자화를 적용하면 1.0 bit/pixel 이하에서도 좋은 畫質을 얻을 수 있다. 그러나, 畫像 블록의 크기가 커질수록 코드벡터의 數는 指數的으로 增加하여, 탐색하는데 필요한 계산량이 막대하다.

코드북 구성에 널리 쓰이는 LBG 알고리듬은 계산상의 冗長度(redundancy)가 크므로 코드북을 구성하는데 많은 시간이 소요된다. W. Equitz[3][4]는 k-d tr-

*準會員, **正會員, 서울大學校 電子工學科
(Dept. of Elec. Eng., Seoul Nat'l Univ.)

接受日字 : 1989年 6月 1日

ee와 Pairwise Nearest Neighbor(PNN) 알고리듬을 이용하여 LBG 알고리듬에 비해 소요시간을 평균 1/20 정도로 줄였다. 그러나 이 알고리듬은 k-d tree를 잡는 방법에 따라 소요시간 및 성능의 차이가 심하고 LBG 알고리듬보다 성능이 떨어지므로 LBG 알고리듬 만큼 널리 쓰이지 않고 있다.

본 논문에서는 효율적으로 초기 코드북을 예측하기 위한 이진 분리(binary splitting) 방법을 제안한다. 또한 구분화(partition)를 위한 탐색(searching)을, LBG 알고리듬에서 처럼 모든 코드 벡터들에 대해 하지 않고, 이진분리 방법으로 초기 코드북을 구성한 후 반복적 최적화(iterative optimization) 알고리듬[3]으로 트레이닝 벡터의 인덱스에 해당하는 코드벡터와 자승 오차(squared error)가 작은 몇 개의 코드벡터들에 대해서만 探索을 함으로써, 소요시간을 줄이는 알고리듬을 提案한다. 이진 분리 방법의 결과를 이용하면 초기 코드벡터들 상호간의 자승 오차가 작은 순서를 整列(sorting) 과정없이 예측할 수 있으므로, 자승 오차가 작은 몇 개의 코드벡터에 대해서만 탐색하면 된다. 이때 탐색하는 코드벡터의 갯수를 변수로 하여 소요시간을 바라는 성능에 맞게 조정할 수 있다. 또한 제안된 알고리듬은 Equitz 의 방법보다 성능이 우수하고 구현이 용이하다.

Ⅱ. 양자화 알고리듬

벡터 量子化는 k 次元 집합 R^k로 부터 M개의 원소를 갖는 R^k의 부분집합 C에로의 函數이다.[1][2] 즉, 함수 Q는 $R: R^k \rightarrow C$이고, $C=\{\bar{y}_i, i=0, 1, 2\cdots, M-1\}$이다. 이때 집합 C를 "코드북"이라고 하고, 코드북을 만들기 위해 필요한 집합 $X(X\subset R^k, X=\{\bar{x}_i, i=0, 1, 2, \cdots, N-1\})$를 "트레이닝 집합(training set)," 각 $\bar{x}_i$를 "트레이닝 벡터"라고 한다. 단, N은 트레이닝 벡터들의 갯수, M은 코드벡터의 갯수이다.

$x\in R^k$일때 널리 쓰이는 양자화 劣化 測定値(distortion measure)는,

$$D=\frac{1}{K}E(\|\bar{x}-Q(\bar{x})\|^2)$$

로 정의되는 平均 自乘 誤差(mean squared error)이다.

코드북은 다음의 조건을 만족하도록 구성된다.[1]

첫째,

$$C_i=\{\bar{x}: \|\bar{x}-\bar{y}_i\|\leq\|\bar{x}-\bar{y}_j\|, i\neq j, 0\leq i, j\leq M-1\} \quad (1)$$

인 M개의 상호 排他的인 클러스터(disjoint clusters)들로 나눈다. 이때 $\bar{x}\in C_i$이면 $Q(\bar{x})=\bar{y}_i$가 된다.

둘째, $\bar{y}_i$는 다시 C_i내에서의 MSE가 최소가 되도록 갱신한다. 즉,

$$\bar{y}_i=\frac{1}{n_i}\sum_{\bar{x}\in C_i}\bar{x}=\mathrm{Cent}(C_i) \quad (2)$$

이고, 이것은 클러스터 C_i의 圓心(centroid)이다. 단, n_i는 C_i원소들의 갯수이다.

J. Max[5], S. P. Lloyd[6]에는 원신호의 확률밀도함수를 알때 양자화기(quantizer)가 전체적인 最適化가 되기 위한 2가지 필요조건이 제시되어 있다.[1]에서는 S. P. Lloyd[6]에 제안된 방법 I을 다차원과 原信號의 확률밀도함수를 모를 경우에 대해 일반화하였다. 따라서 이 알고리듬을 일반화된 Lloyd 알고리듬(generalized Lloyd algorithm), Linde-Buzo-Gray 알고리듬 혹은 LBG 알고리듬이라고 부른다.

전체적인 최적화(globally optimum)와 局部的인 최適化(locally optimum)는 두 양자화기의 미분치가 0이지만, 전체적인 최적화는 양자화기의 평균 양자화 잡음이 최소가 될때를 말하고, 국부적인 최적화는 양자화기를 약간만 변화시키더라도 양자화 잡음이 증가할때를 말한다.[1] LBG 알고리듬은 전체적인 최적화에 이르는 필요충분조건을 만족하는 것이 아니라 필요조건만을 만족하는 것이므로 이 알고리듬으로 구성된 양자화기가 국부적인 최적화에는 이를 수 있지만 항상 전체적인 최적화에 이르지는 않는다. 서로 다른 초기 코드북에 따라 LBG 알고리듬이 수렴한 이후 서로 다른 국부적인 최적화에 이르게 되므로 수렴한 이후의 성능도 달라진다.

Ⅲ. 二進 分理 方法

二進 分理(binary splitting) 방법은 초기 1개의 코드벡터에서 시작하여 원하는 코드벡터의 갯수가 될 때까지 코드벡터의 갯수를 2배씩 늘려가는 알고리듬이다. 이진 분리방법은 다음과 같다. n_i는 C_i의 원소 갯수를 말한다.

벡터 차원이 k일때 $\bar{\varepsilon}_i\cdot\bar{\varepsilon}_j=0 (i\neq j, 1\leq i, j\leq k)$가 되도록 k개의 분리 벡터(splitting vector)를 정한다. 이때 $\|\bar{\varepsilon}\|$는 어떤 값도 무방하다.

[step 1] 초기 m=1이다. 즉, 초기 클러스터의 갯수는 1이다. $C_0'=X$, $\bar{y}_0'=\mathrm{Cent}(C_0')$로 한다. X는 트레이닝 집합이다. 분리 벡터의 인덱스 r을 r=1로 한다.

[step 2] m개의 클러스터를 다음과 같이 분리한다. 각 클러스터의 원심이 $\bar{y}_i'$이고, 분리 벡터

$\bar{\varepsilon}_r$에 대해 $\bar{y}_i'$를 $\bar{y}_{2i}=\bar{y}_i'+\bar{\varepsilon}_r$, $\bar{y}_{2i+1}=\bar{y}_i'-\bar{\varepsilon}_r$로 놓고 C_i'를 다음과 같이 분리한다.

$C_{2i}=\{\bar{x}:\bar{x}\in C_i', \|(\bar{y}_i'+\bar{\varepsilon}_r)-\bar{x}\|\leqq\|(\bar{y}_i'-\bar{\varepsilon}_r)-\bar{x}\|\}$, C_{2i}의 분리벡터 인덱스는 r이다.

$C_{2i+1}=\{\bar{x}:\bar{x}\in C_i', \|(\bar{y}_i'-\bar{\varepsilon}_r)-\bar{x}\|<\|(\bar{y}_i'+\bar{\varepsilon}_r)-\bar{x}\|\}$, C_{2i+1}의 분리 인덱스는 r이다.

n_{2i} 혹은 n_{2i+1}이 0이면 C_i'의 분리벡터 인덱스를 $r\leftarrow r+1$로 하고 위와 같이 C_i'를 분리한다. $1\leqq r\leqq k$이다.

[step 3] $y_{2i}=\mathrm{Cent}(C_{2i})$, $y_{2i+1}=\mathrm{Cent}(C_{2i+1})$ $(0\leqq i\leqq m-1)$로 갱신한다. 이때 클러스터의 갯수는 2m이 된다.

[step 4] 2m=M이면 중단한다.
그렇지 않으면 $C_j\rightarrow C_j'$, $y_j\rightarrow y_j'$ $(0\leqq j\leqq 2m-1)$, $2m\rightarrow m$으로 하고 step2, step3을 반복한다.

분리 벡터 $\bar{\varepsilon}_1$방향으로 클러스터를 분리하는 과정에서 C_{2i} 혹은 C_{2i+1}의 원소 갯수가 0이면 C_i'를 $\bar{\varepsilon}_1$방향으로 분리할 수 없으므로 $\bar{\varepsilon}_1$에 수직인 $\bar{\varepsilon}_2$방향으로 분리하는 것이 효율적이다. 그 다음 C_{2i}와 C_{2i+1}은 $\bar{\varepsilon}_2$방향으로 분리한다. $\bar{\varepsilon}_2$방향으로 분리하려는 클러스터들이 $\bar{\varepsilon}_2$방향으로 분리 되지 않을때는 $\bar{\varepsilon}_1$, $\bar{\varepsilon}_2$에 수직인 $\bar{\varepsilon}_3$방향으로 분리한다. 이와 같은 방법으로 2m=M이 될때까지 반복한다.

이진 분리 알고리듬을 그림으로 살펴보면 그림1과 같다. 그림1에서 알 수 있듯이 이진 분리 알고리듬은 방향성을 가지고 공간을 구분화한다.

LBG 알고리듬의 1회당 소요시간과 이진분리 방법의 소요시간을 비교하면 다음과 같다. 계산시간에 있어서 곱셈계산과 덧셈계산 중 곱셈계산에 소요되는 시간이 지배적이므로 전체계산 시간을 곱셈계산만으로 近似할 수 있다. 입력된 트레이닝 벡터 $\bar{x}$와1개의 코드벡터$\bar{y}$와의 자승 오차 $\|\bar{x}-\bar{y}\|^2$를 계산하는 시간을 $\triangle t$, 코드벡터의 갯수를 $M=2^z$라고 가정하자. 트레이닝 벡터의 갯수를 N이라고 할 때, LBG 알고리듬의 1회당 소요시간은 약 $M\cdot\triangle t\cdot N$이다. 이진분리 방법은 각 트레이닝 벡터가 z회씩 2개의 코드벡터와 自乘誤差를 계산하는 것이므로 소요시간은 약 $2\cdot z\cdot\triangle t\cdot N$이다. 예를 들어 $M=512=2^9$ 일때 z=9이므로 이진분리 방법은 LBG 알고리듬의 1회당 소요시간의 약 2·9/512에 불과하다.

그림2는 1차원 공간에서 二進 分理 방법으로 초기 코드북으로 구성했을때 각 코드 벡터 상호간의 위치 및 인덱스를 나타낸 것이다. 그림3은 16차원 공간에서 이진분리 방법으로 구성된 초기 코드북에서 $\bar{y}_0$, $\bar{y}_{240}$과 가까운 코드벡터의 순으로 코드벡터의 인덱스를 나열한 것이다. 그림1, 그림2, 그림3에서 인덱스

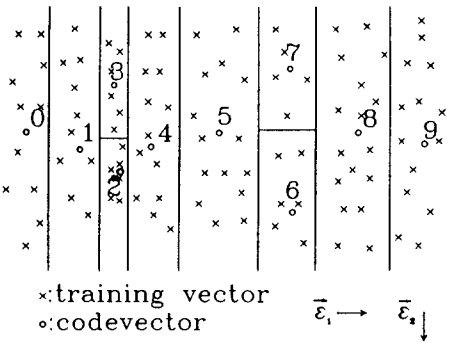

그림 1. 이진 분리 방법에 의한 구분화(2차원)
Fig. 1. Partition by binary splitting method (2 dimension).

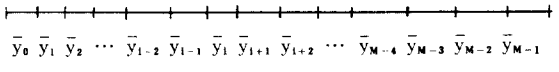

그림 2. 이진 분리 방법으로 구성한 초기 코드벡터들 상호간의 위치(1차원)
Fig. 2. Position of initial codevectors generated by binary splitting method (1 dimension).

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59

240 239 241 238 243 244 237 245 246 242 235 247 233 236 234 248 249 232 231 250
230 251 228 252 253 227 226 229 225 224 254 223 222 221 255 220 219 218 217 216
215 214 213 212 210 211 209 208 206 207 205 204 203 202 201 200 199 198 197 196

그림 3. 초기 코드벡터 $\bar{y}_0$, $\bar{y}_{240}$과 가까운 순서로 정렬된 코드벡터들의 인덱스(16차원)
Fig. 3. Indices of codevectors sorted out in order near to initial codevectors $\bar{y}_0$, $\bar{y}_{240}$ (16 dimension).

차가 작은 코드벡터들 간의 거리가 인덱스 차가 큰 코드벡터들간의 거리보다 근사적으로 가까움을 알 수 있다.

LBG 알고리듬을 수행할 경우 입력된 트레이닝 벡터$\bar{x}$가 초기 C_i의 원소이고, $\bar{y}_j^*$ $(i=j^*)$와 가장 가깝다면 $\bar{x}$를 C_j^*로 옮겨야 한다. 1차원의 경우 가장 가까운 $\bar{y}_j^*$는 $\bar{y}_{i-1}$, $\bar{y}_{i+1}$ 중 하나일 것이다. 그림 1에서 알 수 있듯이 다차원의 경우는 1차원의 경우와 다르다. 그러나 $\bar{y}_j^*$는 $\bar{y}_i$ 근처의 코드벡터일 것이고, 그림 1.

그림2, 그림3에서 알 수 있듯이 i와 인덱스 차가 작은 코드벡터일 것이다. 초기 $x\in C_i$, $s\leqq M-1$을 만족히 크다면,

$$\min_{|j-i|\leqq s} \|\bar{x}-\bar{y}_j\| \leqq \min_{|k-i|>s} \|\bar{x}-\bar{y}_k\|$$

이다. 즉, s가 충분히 크다면 초기 인덱스가 i인 트레이닝 벡터는 $\bar{y}_{i-s}$, $\bar{y}_{i-s+1}$, …, $\bar{y}_i$, …, $\bar{y}_{i+s-1}\bar{y}_{i+s}$들과의 최소거리가 $\bar{y}_0, \bar{y}_1, \cdots, \bar{y}_{i-s-1}\bar{y}_{i+s+1}, \cdots, \bar{y}_{M-1}$ 코드벡터들과의 최소거리보다 가깝게 된다. 따라서 LBG 알고리듬에서 트레이닝 벡터$\bar{x}$에 대해 $\bar{y}_i$를 중심으로 $\bar{y}_i$와 가까운 몇개의 코드벡터들만 탐색하면 되는데, $\bar{y}_i$와 가까운 코드벡터들은 인덱스가 i와 인덱스 차가 작은 코드벡터들이므로 인덱스 차가 작은 몇개의 코드벡터들만 探索한다. 이때 탐색하는 범위를 다음과 같이 정의한다.

$$I_i=\{j: |j-i|\leqq S\} \qquad (3)$$

를 만족하는 인덱스 집합I_i를 인덱스가 i인 트레이닝 벡터의 探索區間(searching range)이라고 정의한다. 탐색구간내에서만 탐색하는 것을 部分探索이라고 정의할 수 있다. 그림4는 탐색구간 I_i를 나타낸 것이다.

$$\underbrace{\bar{y}_0\ \bar{y}_1 \cdots \bar{y}_{i-s}\ \bar{y}_{-s+1} \cdots \bar{y}_{i-1}\ \bar{y}_i\ \bar{y}_{i+1} \cdots \bar{y}_{i+s-1}\ \bar{y}_{i+s} \cdots \bar{y}_{M-2}\ \bar{y}_{M-1}}_{\text{searching range}=2S+1}$$

그림 4. 인덱스가 i인 트레이닝 벡터의 탐색구간
Fig. 4. Searching range of training vector of which index is i.

LBG 알고리듬에서 부분탐색을 할 경우 탐색구간내에 j^*를 찾지 못하더라도 자승평균 오차를 줄일 수 있는 $\hat{j}$를 찾으므로, 전구간 탐색때보다 성능이 크게 떨어지지 않는 수준에서 소요시간을 줄일 수는 있다. 그러나 전구간 탐색때보다 성능이 떨어질 수 밖에 없으므로 소요시간은 줄어들지만 좋은 방법이라고 할 수가 없다. 전구간 탐색을 했을때 성능이 LBG 알고리듬보다 좋은 알고리듬이라면 부분탐색으로 성능이 떨어지더라도 LBG 알고리듬보다 성능이 떨어지지 않는 수준에서 소요시간을 줄일 수 있을 것이다. 다음절에서는 이러한 조건을 만족시키는 反復的 최적화(iterative optimization)알고리듬을 설명한다.

Ⅳ. 반복적 최적화(iterative optimization)알고리듬[3]

트레이닝 벡터 $\bar{x}(\bar{x}\in X)$의 갯수를 N으로 하고, 초기 코드북의 코드벡터$\bar{y}_i$ $(i=0, 1, 2, \cdots, M-1)$들의 값 및 트레이닝 벡터들의 인덱스를 임의로 정한다.

양자화 잡음의 기준이 되는 (4)식의 자승 오차 합을 다음과 같이 나타낼 수 있다.

$$J_e=\sum_{i=0}^{M-1} J_i \qquad (4)$$

J_i는 클러스터 C_i의 자승 오차 합으로써

$$J_i=\sum_{\bar{x}\in Ci} \|\bar{x}-\bar{y}_i\|^2 \qquad (5)$$

이다. 만약 트레이닝 벡터$\bar{x}'$가 현재 클러스터 C_i의 원소인데, 다른 클러스터 C_j로 옮겼다면 코드벡터 $\bar{y}_j$는 다음과 같이 바뀐다.

$$\bar{y}=\frac{\sum_{\bar{x}\in C_j}\bar{x}+\bar{x}'}{n_j+1}=\frac{\sum_{\bar{x}\in C_j}\bar{x}}{n_j+1}+\frac{\bar{x}'}{n_j+1}$$

$$=\frac{(\sum_{\bar{x}\in C_j}\bar{x})n_j}{n_j(n_j+1)}+\frac{\bar{x}'}{n_j+1}=\bar{y}_j\frac{n_j}{n_j+1}+\frac{\bar{x}'}{n_j+1}$$

$$=\bar{y}_j+\frac{\bar{x}'-\bar{y}_j}{n_j+1} \qquad (6)$$

이 된다. 또한, J_j는 다음과 같이 바뀐다.

$$J_j'=\{\sum_{\bar{x}\in C_j}\|\bar{x}-\bar{y}_j'\|^2\}+\|\bar{x}'-\bar{y}_j'\|^2$$

$$=\{\sum_{\bar{x}\in C_j}\|\bar{x}-\bar{y}_j-\frac{\bar{x}'-\bar{y}_j}{n_j+1}\|^2\}+\|\bar{x}'-\bar{y}_j-\frac{\bar{x}'-\bar{y}_j}{n_j+1}\|^2$$

$$=\{\sum_{\bar{x}\in C_j}\{\|\bar{x}-\bar{y}_j\|^2-2(\bar{x}-\bar{y}_j)^t\frac{(\bar{x}'-\bar{y}_j)}{n_j+1}+(\frac{1}{(n_j+1)}\|\bar{x}'-\bar{y}\|)^2\}\}$$

$$+\|\frac{n_j}{n_j+1}(\bar{x}'-\bar{y}_j)\|^2$$

$$=\{\sum_{\bar{x}\in C_j}\|\bar{x}-\bar{y}_j\|^2\}-2\frac{(\bar{x}'-\bar{y}_j)^t}{n_j+1}\sum_{\bar{x}\in C_j}(\bar{x}-\bar{y}_j)+\frac{n_j}{n_j+1}\|\bar{x}'-\bar{y}_j\|^2 \qquad (7)$$

그런데, $\frac{\sum_{\bar{x}\in C_j}\bar{x}}{n_j}=\bar{y}_j$이므로 식(7)은

$$J_j'=J_j+\frac{n_j}{n_j+1}\|\bar{x}'-\bar{y}_j\|^2 \qquad (8)$$

이 된다.

$n_i>1$이면 같은 방법으로

$$\bar{y}_i'=\bar{y}_i-\frac{\bar{x}'-\bar{y}_i}{n_i-1}, \qquad (9)$$

$$J_i'=J_i-\frac{n_i}{n_i-1}\|\bar{x}'-\bar{y}_i\|^2 \qquad (10)$$

이 된다.

(8), (10)에서 $\triangle J_j$, $\triangle J_i$를

$$\triangle J_j = \frac{n_j}{n_j+1}\|\bar{x}'-\bar{y}_j\|^2 \quad (11)$$

$$\triangle J_i = \frac{n_i}{n_i-1}\|\bar{x}'-\bar{y}_i\|^2 \quad (12)$$

로 정의된다. $\triangle J_j$, $\triangle J_i$를 "가중 자승 오차(weighted squared error)"라고 부른다.

식(11), (12)를 이용하면, $\bar{x}'$를 클러스터 C_{j*}로 옮겼을 때 J_e를 최대로 減少시키는 클러스터 C_{j*}를 간단히찾을 수 있고, 식(6), (9)을 이용하여 $\bar{y}_{j*}$와 $\bar{y}_i$를 간단히 更新할 수 있다.

$$\triangle J_i > \min_{\text{all } j} \triangle J_j (j\neq i, j=0,\cdots,M-1) = \triangle J_j \quad (13)$$

이면, 클러스터 C_{j*}로 트레이닝 벡터 $\bar{x}'$를 옮기고,

$$\triangle J_i \leqq \min_{\text{all } j} \triangle J_j (j\neq i, j=0,\cdots,M-1) \quad (14)$$

이면, 옮기지 않는다.

(13)을 만족하는 클러스터로 트레이닝 벡터를 옮기면,

$$J_e' = J_e - \triangle J_i + \triangle J_{j*} \quad (15)$$

이 되어 자승 오차 합 J_e는 $\triangle J_i - \triangle J_{j*}$만큼 감소한다.

$\bar{x}'$를 옮기게 되면, 클러스터C_i와 클러스터 C_{j*}의 $\bar{y}_i$, $\bar{y}_{j*}$를 식(6), (9)에 따라 更新하고, n_i, n_{j*}및 $\bar{x}'$의 인덱스를 修正한다.

$n_i=1$이면 (13)을 만족하는 j^*가 존재하여 트레이닝 벡터를 옮길 경우 클러스터가 없어지게 되므로 옮기지 않는다.

반복적 최적화 알고리듬은 초기 코드북에 따라 수렴한 후의 성능차가 심하다. 그러나 초기 코드북을 二進 分理 방법으로 구성하면 동 떨어진 코드벡터가 없고 코드벡터가 트레이닝 벡터들의 내부에 적절히 분포되어 있을 경우는 LBG 알고리듬 보다 성능이 좋다.

V. 고속 알고리듬

이진 분리 방법으로 초기 코드북을 구성한 후 반복적 최적화 알고리듬을 수행하는 경우를 가정하자. 일반적으로 $N \gg M$, $N/M \gg 1$이고 $n_i \cong n_j \cong N/M$ 이므로, $n_j/(n_j+1)$과 $n_i/(n_i-1)$은 $n_j/(n_j+1) \cong n_i/(n_i-1) \cong 1$ 이 된다. 그 결과 $\triangle J_i \cong \|\bar{x}'-\bar{y}_i\|^2$, $\triangle J_j \cong \|\bar{x}'-\bar{y}_j\|^2$ 이다. 따라서 LBG 알고리듬에서 처럼 탐색구간을 (3)과 같이 예측할 수 있다.

반복적 최적화 알고리듬에 부분탐색을 이용한 방법 I은 다음과 같다.

[step 1] N개의 트레이닝 벡터들을 이용하여 이진 분리방법으로 초기 코드북을 구성하고, n_i, J_i $(i=0, 1, 2, \cdots M-1)$을 구한다. $s\,(0\leqq s\leqq M-1)$를 결정한다.

[step 2] $t=1$으로 하고, 임계치 θ를 정한다. $J_e(t)$를 계산한다.

[step 3] 트레이닝 벡터$\bar{x}_k\,(k=0, 1, 2, \cdots, N-1)$에 대해 차례로 step4, step5를 반복한다.

[step 4] $\bar{x}_k$가 속한 클러스터 C_i의 원소 갯수 $n_i>1$ 이면, $\triangle Ji$, $\triangle J_j\,(j=i, j=0, 1, 2, \cdots, M-1)$를 계산하고,
$n_i=1$이면 다음 트레이닝 벡터에 대해 step 4를 수행한다.

[step 5] $\triangle J_i > \min_{|i-j|\leqq s} \triangle J_j$이면, $\min_{|i-j|\leqq s} \triangle J_j$인 클러스터를 클러스터 C_j라 하고, $\bar{x}_k$를 클러스터 D_j로 옮기고, $\bar{y}_j$, $\bar{y}_i$를 식(6), (9)에 따라, J_j, J_i를 (8), (10)에 따라 갱신한다.
$n_i \leftarrow n_i-1$, $n_j \leftarrow n_j+1$로 하고, $\bar{x}_k$의 인덱스를 j로 한다.
$\triangle J_i \leqq \min_{|i-j|\leqq s} \triangle J_j$이면, 옮기지 않는다.

[step 6] $\gamma = \dfrac{J_e(t)-J_e(t+1)}{J_e(t)} \leqq \theta$이면 중단하고, 그렇지 않으면 $t \rightarrow t+1$, step3으로 가서 다시 수행한다.

위의 알고리듬은 M개의 코드벡터 중 $2s+1$개를 탐색하게 되고 가중 자승 오차를 계산할때는 자승오차를 계산할때보다 곱셈이 1회 많으므로 1회당 소요시간은 약 $(2s+1)\cdot((k+1)/k)\cdot N\cdot\triangle t$가 된다.

위의 알고리듬을 반복할 경우 인덱스와 코드벡터 상호간의 거리가 그림1, 그림2, 그림3과 같은 관계로 유지되지 않는다. 결국 반복 횟수가 거듭될수록 탐색구간 예측 오차가 커지는 것이므로 탐색구간을 크게 함으로써 오차를 보상하여야 한다. 또한 변화율 γ를 큰 값으로 유지하기 위해 $1/\gamma$을 탐색구간에 고려한 방법 II는 다음과 같다.

방법I에서 step 1, step6만 다음과 같이 수정하고 나머지는 방법 I 과 같다.

[step 1] N개의 트레이닝 벡터들을 이용하여 이진 분리방법으로 초기 코드북을 구성하고, $n_i, J_i\,(i=0, 1, 2, \cdots, M-1)$을 구한다. 초기 $s=1$ 및 α, β, MAX, T를 결정한다.

[step 6] $\gamma = \dfrac{J_e(t)-J_e(t+1)}{J_e(t)}$, $t<T$이면, $t \rightarrow t+1$, $s \leftarrow s+\alpha\cdot 1/\gamma+\beta$로 하고

s>MAX 이면 s=MAX로 한다.
그렇지 않으면 중단한다.

Ⅵ. 실험 및 결과

實驗은 256×256 크기의 국제 표준 畵像인 LENA와 HOUSE에 대하여 IBM PC 386 기종으로 컴퓨터 시뮬레이션(computer simulation)을 하였다. 벡터의 크기는 4×4, 臨界値 θ는 0.001로 하였다. 트레이닝 집합은 내부 트레이닝 집합(inside training set)으로 하였다.

표 1. 각 알고리듬의 성능
Table 1. Performances of respective algorithms.

방법 / 화상	코드북 크기	LBG		제안방법Ⅰ 전구간		제안방법Ⅰ s=7		제안방법Ⅱ $\alpha=0.3, \beta=2$		Equitz
		반복 횟수	PSNR	반복 횟수	PSNR	반복 횟수	PSNR	반복 횟수	PSNR	PSNR
LENA	256	14	28.8	11	29.6	11	28.9	4	28.8	28.3
	512	11	30.0	12	31.5	10	30.6	4	30.2	29.4
HOUSE	256	15	31.9	6	32.5	7	31.8	4	31.7	31.2
	512	9	32.4	10	33.0	9	33.3	4	33.2	31.8

PSNR : dB

표 2. 각 알고리듬의 소요시간
Table 2. Computation times of respective algorithms.

방법 / 화상	코드북 크기	LBG	제안방법Ⅰ 전구간	제안방법Ⅰ s=7	제안방법Ⅱ $\alpha=0.3, \beta=2$	Equitz
LENA	256	6253	4906	297	107	339
	512	9593	10525	317	118	320
HOUSE	256	6697	7123	305	107	352
	512	7857	8796	290	118	329

시간 : sec

표 3. 각 알고리듬의 소요시간비
Table 3. Ratios of respective algorithms.

방법 / 화상	코드북 크기	제안방법Ⅰ 전구간	제안방법Ⅰ s=7	제안방법Ⅱ $\alpha=0.3, \beta=2$	Equitz
LENA	256	1/1.27	1/21.2	1/58.44	1/18.45
	512	1.07	1/30.26	1/81.30	1/29.98
HOUSE	256	1.06	1/21.96	1/62.59	1/19.03
	512	1.12	1/27.09	1/74.54	1/23.38

비 : $\frac{\text{제안된 알고리듬의 소요시간}}{\text{LBG 알고리듬의 소요시간}}$

표1은 각 알고리듬의 반복 횟수와 PSNR[7]을 나타낸 것이다. LBG 알고리듬은 초기 코드북을 임의로 정했다. 제안방법Ⅰ은 탐색구간이 15이다. 제안방법Ⅱ는 반복 횟수를 4회로 고정시켰다. 그 이유는 반복 횟수가 4회를 넘어서면 변화율 γ가 0.1이하로 떨어지고 4회만 반복하더라도 PSNR이 LBG 알고리듬의 PSNR 보다 높기 때문이다. 반복적 최적화 알고리듬은 초기 코드북에 따라 PSNR의 차가 심하다. 또한 트레이닝 벡터가 입력되는 순서에 의해서도 PSNR의 차가 생긴다. LBG 알고리듬은 초기 코드북에 따라 수렴한 후의 PSNR이 다를 수는 있지만 그 차가 크지는 않다. 표1과 같은 실험 조건에서는 0.3dB 이상의 차가 나지 않지만, 반복적 최적화 알고리듬에서는 3dB 이상의 차가 생길 수 있다. 그러나 Ⅳ절에서 설명한 바와 같이 초기 코드북을 이진 분리 방법으로 정하면 코드벡터들이 적절히 분포되어 있으므로 표1과 같이 수렴 후의 PSNR이 LBG 알고리듬의 PSNR 보다 높다. 반복적 최적화 알고리듬의 PSNR이 LBG 알고리듬의 PSNR 보다 거의 1dB 이상 높다. Ⅱ절에서 설명한 바와 같이 LBG 알고리듬이 만족하는 조건 (1), (2)는 전체적인 최적화를 만족하기 위한 필요조건이다. 반면 반복적 최적화 알고리듬은 트레이닝 벡터를 직접적으로 M 개의 클러스터 중 전체 자승 오차를 최대한 줄일 수 있는 클러스터로 이동시키기 때문에 PSNR 면에서는 LBG 알고리듬을 앞선다. 제안방법Ⅰ에서 탐색구간이 전구간보다 작고 15보다 클 경우는 전구간 탐색때보다 PSNR은 떨어지지만 LBG 알고리듬의 PSNR보다는 떨어지지는 않았다. 이것은 제안방법Ⅰ의 부분탐색의 유용성을 입증한다. LBG 알고리듬의 경우 초기 코드북에 따라 반복횟수가 달라진다.[4] 그런데 임계치 θ를 0.01로 잡을 경우 제안방법Ⅰ의 전구간 탐색과 LBG 알고리듬의 반복 횟수는 코드북 크기가 256일때는 7회, 코드북 크기가 512 일때는 5회 정도로 일정했다.

그러나 변화율이 0.01 이하로 떨어지고 난 이후 0.001로 떨어질때까지 반복횟수는 화상에 따라 다르다. 또한 전체 반복횟수의 1/2 정도를 차지하면서도 0.1 dB 정도밖에 PSNR이 올라가지 않는다.

Equitz[4]의 방법은 표1에서 알 수 있는 바와 PSNR이 LBG 알고리듬보다 0.5dB 정도 떨어졌다. 제안방법Ⅱ는 반복횟수를 4회로 고정시켰다. 그 이유는 반복횟수가 4회를 넘어서면 변화율 γ는 0.1이하로 떨어져서 변화율이 0.001 이하가 될때까지 반복하더라도 PSNR은 0.2~0.3dB 밖에 향상되지 않고, $\alpha=0.3$ $\beta=2$로 하였을 때 4회만 반복하더라도 PSNR이 L

BG 알고리듬의 PSNR 보다 높기 때문이다. α와 β는 클수록 탐색구간의 증가폭이 큰 것이므로 PSNR은 높다.

표2, 표3은 각 알고리듬의 소요시간 및 소요시간비를 나타낸 것이다. 이진분리 방법의 소요시간은 코드북 크기가 256 일때 37초, 코드북의 크기가 512 일때는 42초였다. 제안방법 I, II에서는 이진 분리에 소요되는 시간을 전체 소요시간에 포함시켰다. Equitz의 방법은 코드벡터의 갯수를 트레이닝 벡터 갯수 N개에서 M개가 될때까지 줄이는 것이므로 코드벡터의 갯수가 클수록 소요시간이 짧다. 위에서 서술한 바와 같이 변화율이 0.01에서 0.001 이하로 떨어지더라도 PSNR은 0.1dB 정도밖에 높아지지 않는다. 변화율의 임계치를 0.01로 LBG 알고리듬의 소요시간은 1/2로 줄어든다. LBG 알고리듬에 대한 Equitz 방법의 소요시간비는 2배 정도로 커진다. 그러나 제안된 방법 I 에서는 임계치를 0.01로 하더라도 반복횟수는 1/2 정도 떨어지므로 시간비는 줄지 않는다. 표3은 제안된 방법 I, II에서 코드북이 커진 만큼 탐색구간을 크게 할 필요가 없음을 보여준다.

Ⅶ. 결　　론

제안된 방법은 중복 계산량이 많은 LBG 알고리듬 대신, 반복적 최적화 알고리듬을 기초로한 선택적인 探索作業을 도입하여, 계산량을 줄임으로써 소요시간을 1/60 이하로 감소시켰다. 이상과 같은 알고리듬을 이용하면, 探索區間을 변수로 하여 LBG 알고리듬보다 PSNR이 떨어지지 않는 수준에서 소요시간을 調整할 수 있었다. 또한 소요시간에서 잇점은 없지만 반복적 최적화 알고리듬으로 양자화기를 구성하면 LBG 알고리듬보다 성능이 우수한 양자화기를 구성할 수 있음을 보였다. 그리고 초기 코드북을 효율적으로 구성할 수 있는 二進 分理 방법을 제안하였다. 본 논문의 알고리듬은 W. Equitz의 방법보다 PSNR이 높음을 보였고, 구현이 간편하다.

參 考 文 獻

[1] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantization design," *IEEE Trans. Comm.*, vol. COM-28, Jan. 1980.

[2] R.M. Gray, "Vector quantization," *IEEE ASSP Mag.*, pp. 4-28, Apr. 1984.

[3] R. O. Duda and P. E. Hart, *Pattern Classification and Scence Analysis.* NY: Wiley, 1971.

[4] W.H. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, pp. 1568-1575, Oct. 1989.

[5] J. Max, "Quantizing for minimum distortion," *IRE Trans. Inform. Theory*, vol. IT-6, pp. 7-12, Mar. 1960.

[6] S.P. Lloyd, "Least square quantization in PCM," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 129-137, Mar. 1982.

[7] W.K. Pratt, *Digital Image Processing.* NY: Willy, 1978.

著 者 紹 介

李 周 熙(準會員)

1965年 5月 2日生. 1988年 서울대학교 전자공학과 졸업. 1988年 3月～현재 서울대학교 대학원 전자공학과 석사과정 재학중. 주관심분야는 통신 방식, 화상처리 및 코딩 등임.

●

李 忠 雄 (正會員) 第26卷 第5號 參照
현재 서울대학교 전자공학과 교수

丁 海 默(正會員)

1962年 10月 9日生. 1985年 서울대학교 전자공학과 졸업. 1987年 서울대학교 대학원 전자공학과 졸업. 공학 석사학위 취득. 1987年 3月～현재 서울대학교 대학원 전자공학과 박사과정 재학중. 주관심분야는 통신 방식, 화상처리 및 코딩과 HDTV 처리기술 등임.