고속 VQ 부호화 알고리즘

A Fast VQ Encoding Algorithm

백 성 준*, 이 대 룡*, 전 범 기*, 성 굉 모**
(Seongjoon Baek*, Daeryong Lee*, Bumki Jeon*, Koengmo Sung**)

※본 연구는 정보통신 기초연구사업의 지원으로 이루어졌습니다.

요 약

본 논문에서는 새로운 고속 VQ 부호화 알고리즘을 제안한다. 제안된 알고리즘은 최단거리를 가질 수 없는 코드워드와의 거리계산을 피하기 위해 벡터의 두가지 특성, 평균 및 분산을 이용해서 계산량을 줄이고 있다. 제안된 알고리즘은 코드워드와 입력벡터사이의 거리의 기하학적인 관계를 이용해서 최단거리를 갖을 수 없는 코드워드들과의 거리계산을 피하고 있기 때문에 전체 탐색법과 동일한 결과를 가져다 준다. 실험결과는 제안된 알고리즘이 효과적임을 확인시켜준다.

ABSTRACT

In this paper, we present a new fast VQ encoding algorithm. The proposed algorithm facilitates two characteristics of a vector, i.e., mean and variance to reject many unlikely codewords and save a lot of computation time. Since the proposed algorithm, which is based upon geometric considerations, rejects those codewords that are impossible to be the closest codeword, it provides the same results as a conventional exhaustive(or full) search algorithm. The simulation results confirm the effectiveness of the proposed algorithm.

Ⅰ. 서 론

벡터양자화(이하 VQ)는 데이터 압축에 매우 효율적인 방법이며 벡터의 차수가 증가함에 따라 rate-distortion 한계에 접근하는 코딩 성능을 지닌다[1]. 부호화 단계는 코드북 Y중에서 입력 x에 가장 가까운 코드워드를 찾는 과정으로 코드북의 크기가 N이라고 한다면 전체 탐색을 이용할 경우에 N번의 거리계산을 필요로 한다. 여기서 코드북의 크기 N과 입력벡터의 차수 k 그리고 비트-레이트 r(bits/sample) 사이의 관계식은 $N=2^{kr}$이 되므로 부호화 단계에서 계산량은 k와 r이 증가함에 따라 지수함수적으로 증가하게 된다. 따라서 VQ 부호화 단계에서 계산량을 줄이는 문제는 VQ를 여러 분야에 응용하고자 할 경우에 중요한 문제가 된다. 그에 반해 복호화 단계에서는 전송된 코드북의 인덱스를 이용해서 코드북 테이블의 벡터값을 복원시키기만 하면 되므로 계산량이 문제가 되지 않는다.

이런 이유로 VQ 부호화 단계에서 계산량을 줄이기 위한 연구들이 많이 행해졌는데 그것들은 크게 다음의 두가지로 분류될 수 있다. 첫 번째 접근법은 VQ 부호화시 최단거리에 있는 코드워드를 구하지 않고 근사적인 코드워드를 구하는 방법으로서 코딩성능을 약간 희생하는 대신 부호화에 용이한 구조를 채택하는 것이다. 대표적인 방법으로는 트리구조를 이용해서 코드북을 구성하는 TSVQ, K-d tree 등이 있다[2][3].

두 번째 접근법은 전체탐색법에서와 동일하게 최단거리의 코드워드를 구하면서 계산량을 줄이는 방법이다. 이들 가운데 Bei and Gray에 의해 제시된 부분거리 탐색(partial distance search, PDS) 알고리즘은 매우 간단하면서도 효과적인 방법이다[4]. 이 방법은 거리계산시에 모든 차원의 거리계산을 마친 후 지금까지 찾은 최단거리와 비교하지 않고, 매 차원 거리계산을 마친 후 최단거리와 비교하는 방법이다. 이 방법을 사용하면 비교횟수는 늘어나지만 곱셈 수는 줄게 되어 전체적으로 실행속도가 줄어들게 된다. 이것이 가지는 장점은 부가적인 메모리를 요구하지 않는다는 것이지만 그 계산량의 감소폭이 크지 않아서 보다 효율적인 부호화 방법으로서 fast nearest neighbor search(이하 FNNS) 알고리즘과 벡터의 pro-

*서울대학교 전자공학과 박사과정
**서울대학교 전자공학부 교수
접수일자 : 1996년 12월 27일

jection을 이용하는 알고리즘이 제안되었다. FNNS 알고리즘은 코드워드 사이의 거리에 대해 triangle inequality를 이용해서 거리계산이 불필요한 코드워드들을 거리계산에서 제외시킴으로써 계산량을 줄이고 있으며, 실험결과에 따르면 FNNS 알고리즘은 보통의 전체 탐색법에 비해 큰 계산량의 감소를 보인다. 하지만 이 방법은 전체 탐색법에 비해서 모든 코드워드쌍의 거리를 저장하기 위해 $N(N-1)/2$ 만큼의 부가적인 메모리를 필요로 한다. 따라서 코드북의 크기가 커질수록 메모리 요구량은 제곱비로 증가하게 되어 실응용시에 메모리로 인한 문제가 야기될 수 있다[5].

이에 반해 equal-average nearest neighbor search(ENNS)와 같은 projection 알고리즘은 불필요한 코드워드를 거리계산에서 제외하기 위해 벡터의 평균과 거리에 대한 관계를 이용한다. 이 방법은 FNNS 알고리즘과 동등한 또는 더 많은 계산량의 감소를 보여주는데 이에는 코드워드들의 평균을 저장하기 위한 N개의 메모리만이 부가적으로 요구된다[6]. 하지만 이 방법은 유클리드 거리에만 적용할 수 있으므로 다른 거리를 사용해야 하는 VQ의 경우에는 적절한 변형이 필요하다. ENNS를 확장하여 평균 이외에 분산에 대한 거리 관계식을 이용하는 방법이 제안되었는데 - 본 논문에서는 그것을 EENNS(equal-average equal-variance nearest neighbor search)라 부르기로 한다 - 이 방법은 코드워드의 분산을 저장하기 위한 메모리를 포함하여 부가적인 $2N$개의 메모리를 요구하나 ENNS보다 적은 계산량을 필요로 한다[7].

본 논문에서는 평균과 거리에 대한 관계를 이용하는 ENNS 알고리즘에 분산과 거리와의 관계를 부가적으로 이용하는 EENNS 알고리즘을 검토하고 새로운 고속 VQ 부호화 알고리즘을 제안한다. 제안된 알고리즘은 EENNS와 동일하게 입력 벡터의 평균과 분산을 이용하는데 EENNS 알고리즘에서 사용했던 관계식이 아닌 새로운 관계식을 사용한다. 이 관계식을 사용할 경우 입력벡터에 가장 가까운 코드워드를 찾기 위한 탐색영역이 ENNS 알고리즘과 EENNS 알고리즘의 경우에 비해 줄어드므로 제시된 알고리즘은 보다 향상된 계산량의 감소를 얻을 수 있다.

본 논문의 목차는 다음과 같다. 먼저 Ⅱ장에서 ENNS와 EENNS 알고리즘을 검토하고, Ⅲ장에서 새로운 알고리즘을 제안한다. 제안된 알고리즘의 성능을 비교하기 위한 실험 및 토론은 Ⅳ장에 있으며, Ⅴ장에서는 간단한 결론이 주어져 있다.

Ⅱ. ENNS 알고리즘과 EENNS 알고리즘

이 장에서는 ENNS 알고리즘과 EENNS 알고리즘을 살펴본다. 각 알고리즘을 설명하기 전에 먼저 다음을 정의한다.

정의 1: 입력 벡터 $x=(x_1, x_2, ..., x_k)$의 평균 m_x과 분산 V_x^2는 각각 식 (1)과 (2)와 같이 정의된다.

$$m_x=\frac{1}{k}\sum_{j=1}^{k} x_j \tag{1}$$

$$V_x^2=\sum_{j=1}^{k}(x_j-m_x)^2 \tag{2}$$

정의 2: l을 유클리드 공간상의 직선이라고 할 때 l상의 임의의 점 $p=(p_1, p_2, ..., p_k)$가 $p_1=p_2=...=p_k$을 만족하면 그 공간의 l은 중심축(the central line or the central axis of R^k)이라 부른다. 이때 L_x를 이 중심축 l에 대한 x의 projection point라고 하면 $L_x=(m_x, m_x, ..., m_x)$로 표시할 수 있다.

ENNS는 벡터의 평균을 이용해서 불필요한 거리계산을 하지 않도록 하는데 그 중심논리는 다음 정리로 요약될 수 있다.

정리 1: 먼저 $x=(x_1, x_2, ..., x_k)$는 주어진 입력 벡터이고 거리계산을 하고자 하는 코드워드를 $y_i=(y_{i1}, y_{i2}, ..., y_{ik})$라고 하며, 거리는 유클리드 거리로 정의되었다고 하면 x와 y_i의 평균과 거리 사이에는 식 (3)이 성립한다.

$$d(x, y_i)\geq\sqrt{k}\,|m_x-m_{y_i}| \tag{3}$$

정리 1은 임의의 코드워드 y_i에 대해서 관계식 $k(m_x-m_{y_i})^2\geq d_{\min}^2$이 성립한다면 y_i는 x에 가장 가까운 코드워드일 수 없으므로 거리계산이 불필요하게 된다는 것을 알려준다[6][7]. ENNS 알고리즘은 주어진 입력벡터에 대해서 먼저 평균을 계산한 다음 그 평균과 가장 가까운 평균을 가지는 코드워드를 찾고 이 코드워드로부터 그 다음 코드워드를 탐색한다. 이 탐색과정에서 ENNS 알고리즘은 정리 1을 사용하여 불필요한 코드워드와의 거리계산을 생략하므로 전체적으로는 계산량을 감소시킨다.

EENNS 알고리즘은 다음과 같은 정리를 이용해서 벡터의 평균뿐 아니라 분산도 사용하여 보다 계산량을 줄이고 있다.

정리 2: 먼저 $x=(x_1, x_2, ..., x_k)$는 주어진 입력 벡터이고 거리계산을 하고자 하는 코드워드를 $y_i=(y_{i1}, y_{i2}, ..., y_{ik})$라고 하며, 거리는 유클리드 거리로 정의되었다고 하면 x와 y_i의 분산과 거리 사이에는 식 (4)가 성립한다. 식 (4)에서 V_x와 V_{y_i}는 각각 $d(x, L_x)$과 $d(y_i, L_{y_i})$를 나타낸다.

$$d(x, y_i)\geq|V_x-V_{y_i}| \tag{4}$$

주어진 임의의 코드워드 y_i에 대해서 EENNS는 먼저

ENNS에서와 같이 관계식 $k(m_x-m_{y_i})^2 \geq d^2_{min}$을 사용해서 불필요한 코드워드를 거리계산에서 제외한다. 그후 남은 코드워드에 대해서는 정리 2가 함축하고 있는 관계식, $(V_x-V_{y_i})^2 \geq d^2_{min}$을 만족하는 y_i는 x에 가장 가까운 코드워드일 수 없다는 점을 이용해서 보다 많은 코드워드들을 거리계산으로부터 제외한다[7]. 그림 1은 2 차원 공간에서 ENNS와 EENNS 알고리즘의 코드워드 탐색 영역을 보여준다.

Ⅲ. 제안된 알고리즘

제안된 알고리즘은 EENNS와 동일하게 벡터의 평균과 분산을 이용한다. 하지만 위에서 보았듯이 EENNS 알고리즘에서는 이 두 가지 정리가 각각 독립적으로 사용된다. 만약 이 두 가지 정리가 결합될 수 있다면 탐색영역은 보다 축소될 것이며 결과적으로 더 많은 계산량의 감소를 가져올 수 있을 것이다. 제안된 알고리즘은 이러한 견지에서 평균과 분산 그리고 거리 사이에 성립하는 다음과 같은 정리를 발전시켜 코드워드 탐색영역을 위의 두 가지 알고리즘의 경우보다 제한한다.

정리 3: 먼저 $x=(x_1, x_2, \ldots, x_k)$는 주어진 입력 벡터이고 거리계산을 하고자 하는 코드워드를 $y_i=(y_{i1}, y_{i2}, \ldots, y_{ik})$라고 하며, 거리는 유클리드 거리로 정의되었다고 하면 x와 y_i의 평균, 분산과 거리 사이에는 식 (5)이 성립한다. 식 (5)에서 V_x와 V_{y_i}는 각각 $d(x, L_x)$과 $d(y_i, L_{y_i})$를 나타낸다.

$$d(x, y_i)^2 \geq k(m_x-m_{y_i})^2+(V_x-V_{y_i})^2 \quad (5)$$

증명: 만약 p를 $\overrightarrow{y_iL_{y_i}}$에 대한 입력 벡터 x의 projection point라고 하면 그림 2에서 보이듯 식 (6)과 같은 관계식이 성립된다.

$$d(x, y_i)^2 = d(x, p)^2+d(y_i, p)^2 \quad (6)$$

또한 y_i, L_{y_i}, p는 모두 동일한 직선 상에 있으므로 식 (7)이 성립한다.

$$d(y_i, p) \geq |d(y_i, L_{y_i})-d(p, L_{y_i})| \quad (7)$$

여기서 $d(p, L_{y_i})$는 $\overrightarrow{L_xx}$와 $\frac{\overrightarrow{L_{y_i}y_i}}{|\overrightarrow{L_{y_i}y_i}|}$의 내적에 대한 절대값이므로 식 (8)이 성립한다.

$$d(p, L_{y_i}) \leq |\overrightarrow{L_xx}| = d(x, L_x) \quad (8)$$

식 (7)과 (8)을 결합하면 식 (9)을 얻을 수 있다.

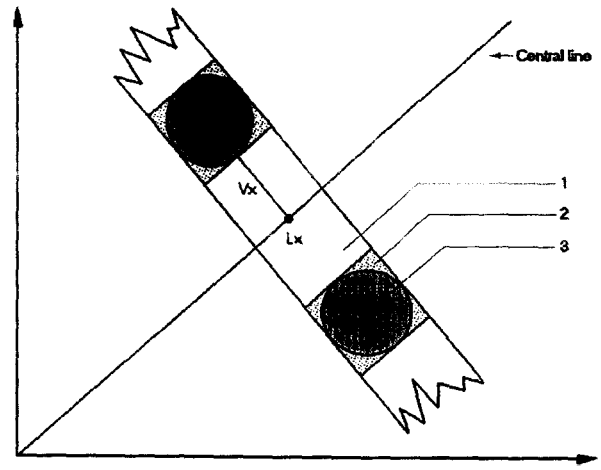


그림 1. 2 차원인 경우에 각 알고리즘의 탐색범위
(1: ENNS 2: EENNS 3: 제안된 알고리즘)

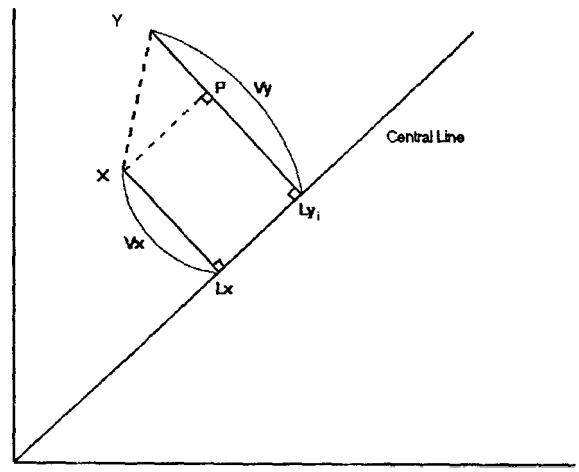


그림 2. 입력벡터와 코드워드와의 관계도

$$d(y_i, p) \geq |d(y_i, L_{y_i})-d(x, L_x)| = |V_{y_i}-V_x| \quad (9)$$

그리고 $d(L_x, L_{y_i})$는 x를 포함하는 평면과 p를 포함하는 평면사이에 가장 가까운 거리이므로 식 (10)이 성립한다.

$$d(x, p)^2 \geq d(L_x, L_{y_i})^2 = k(m_x-m_{y_i})^2 \quad (10)$$

따라서 식 (6), (9)과 (10)를 결합하면 식 (11)을 얻을 수 있다.

$$d(x, y_i)^2 = d(x, p)^2+d(p, y_i)^2 \geq k(m_x-m_{y_i})^2+(V_x-V_{y_i})^2 \quad (11)$$

■

제안된 알고리즘은 계산량의 감소를 위해서 먼저 식 (10)의 결과인 관계식 $k(m_x-m_{y_i})^2 \geq d^2_{min}$를 이용하여 x에 가장 가까울 수 없는 코드워드를 제외하고, 남은 코드워드

들에는 식 (11)의 결과인 $k(m_x-m_{y_i})^2+(V_x-V_{y_i})^2 \geq d^2_{min}$을 이용해서 보다 많은 코드워드들을 거리계산으로부터 제외시킨다. 평균과 분산을 동시에 고려한 제안된 관계식을 하나만 사용하는 대신에 먼저 평균에 대한 관계식을 사용하고 난 다음 제안된 관계식을 사용하는 이유는 분산을 계산하지 않아도 되는 경우에 보다 더 계산량을 줄이기 위해서이다. 식 (11)을 사용하는 경우, 제안된 알고리즘의 코드워드 탐색영역은 그림 1에서 보는 것처럼 ENNS와 EENNS 알고리즘의 탐색영역보다 적어지므로 필요한 계산량은 더욱 감소하게 된다.

제안된 알고리즘의 자세한 절차는 아래에 의사 C 언어로 기술되어 있다. 편의상 k차원 벡터로 이루어진 코드북은 그 평균에 따라 오름차순으로 분류되어 있고, 코드북의 크기는 N이라고 가정한다.

단계 1: 입력 벡터 x의 평균 m_x와 분산 V_x을 구한다.

단계 2: m_x와 가장 가까운 평균을 가진 코드워드 y_i를 이진탐색을 이용해서 구한다.

단계 3: 먼저 $d^2_{min}=d^2(x, y_i)$를 계산하고, $i_c=i_{up}=i_{down}=i$로 놓는다.

단계 4:
```
while(i_up < N-1 or i_down > 0){
    if(i_up < N-1){
        i_up = i_up + 1;   d_E^2 = k(m_x - m_{y_i})^2;
        if(d_E^2 < d_min^2){
            d_V^2 = (V_x - V_{y_i})^2;
            if(d_E^2 + d_V^2 < d_min^2){
                d^2 = d^2(x, y_{i_up});
                if(d^2 < d_min^2)   i_c = i_up, d_min^2 = d^2;
            }
        }
        else   i_up = N-1
    }
    if(i_down > 0){
        i_down = i_down - 1; d_E^2 = k(m_x - m_{y_i})^2;
        if(d_E^2 < d_min^2){
            d_V^2 = (V_x - V_{y_i})^2;
            if(d_E^2 + d_V^2 < d_min^2){
                d^2 = d^2(x, y_{i_down});
                if(d^2 < d_min^2) i_c = i_down, d_min^2 = d^2;
            }
        }
        else   i_down = 0;
    }
}
```

단계 5: 얻어진 i_c를 입력벡터에 가장 가까운 코드워드의 인덱스로 한다.

■

제안된 알고리즘은 부호화 알고리즘 외에 고속 VQ 학습 알고리즘에도 사용 가능한데 이에는 단순 부호화 경우와는 달리 다음 3가지를 고려해야 한다. 그 첫째는 매번 코드북이 갱신될 때마다 그 평균에 따라 코드북을 정렬해야 한다는 점이다. 이 코드북 정렬로 인해 단순 부호화시의 계산량보다 학습시 계산량이 증가하게 되지만 실제는 다음 두 번째와 세 번째 이유로 인해 단순 부호화 때보다 전체적인 계산량이 감소하게 된다. 두 번째는 단순 부호화 때는 매번 데이터의 평균과 분산을 계산해야 하지만 학습할 때에는 첫 번째 부호화 때에만 계산하고 나머지 반복 때에는 이 값을 저장해서 사용할 수 있다는 점이다. 매번 부호화시에 데이터의 평균과 분산을 다시 계산할 필요가 없어지므로 불필요한 오버헤드를 줄일 수

표 1. 평균 거리계산 횟수.

크기	방법	화상			
		Lena	Peppers	Jet	Baboon
128	Full Search	128	128	128	128
	ENNS	7.97	9.05	8.67	26.51
	EENNS	4.26	4.83	4.71	18.72
	Our Method	3.40	3.70	3.58	13.83
256	Full Search	256	256	256	256
	ENNS	14.85	17.56	16.64	51.74
	EENNS	7.13	8.44	8.69	35.65
	Our Method	5.74	3.70	6.79	27.03
512	Full Search	512	512	512	512
	ENNS	27.36	33.89	31.65	100.12
	EENNS	11.73	14.91	14.93	66.63
	Our Method	9.50	11.66	11.68	51.18
1024	Full Search	1024	1024	1024	1024
	ENNS	53.65	67.52	56.74	196.18
	EENNS	25.10	32.51	23.90	115.00
	Our Method	20.42	24.97	18.35	82.40

표 2. 실행속도(단위는 초, 괄호 안은 Full Search Algorithm과 비교한 비율)

크기	방법	화상			
		Lena	Peppers	Jet	Baboon
128	Full Search	15.88	15.93	15.98	15.93
	ENNS	1.49(9.38)	1.59(9.98)	1.60(10.01)	4.17(26.18)
	EENNS	1.15(7.24)	1.21(7.60)	1.21(7.57)	3.51(22.03)
	Our Method	1.09(6.86)	1.10(6.90)	1.10(6.88)	2.91(18.27)
256	Full Search	32.30	32.30	32.35	32.29
	ENNS	2.47(7.65)	2.86(8.85)	2.75(8.50)	8.02(24.84)
	EENNS	1.71(5.29)	1.92(5.94)	1.98(6.12)	6.38(19.76)
	Our Method	1.54(4.77)	1.76(5.45)	1.76(5.44)	5.32(16.48)
512	Full Search	64.54	64.54	64.70	64.54
	ENNS	4.39(6.80)	5.32(8.24)	5.05(7.81)	15.33(23.75)
	EENNS	2.69(4.17)	3.24(5.02)	3.24(5.01)	11.76(18.22)
	Our Method	2.41(3.73)	2.86(4.43)	2.86(4.42)	9.89(15.32)
1024	Full Search	129.19	129.18	129.40	129.07
	ENNS	8.46(6.55)	10.43(8.07)	8.90(6.88)	29.93(23.19)
	EENNS	5.22(4.04)	6.48(5.02)	5.11(3.95)	20.87(16.17)
	Our Method	4.61(3.57)	5.60(4.34)	4.45(3.44)	16.81(13.02)

있어 단순 부호화 때보다 많은 계산량의 감소를 얻을 수 있다. 세 번째는 제일 처음 탐색하는 코드워드를 평균을 이용한 이진 탐색으로 찾는 대신에 대부분의 데이터가 최단 거리 코드워드를 재분류 시에 바꾸지 않으므로 이전 코드워드를 기억하고 이 코드워드로부터 탐색케 한다는 점이다. 이 방법을 사용하게 되면 평균거리 탐색횟수가 크게 줄게 되어 추가적인 계산량의 감소를 얻을 수 있다. 이와 같은 점을 고려하면 제안된 알고리즘은 부호화만이 아니라 VQ 학습에도 효과적으로 적용될 수 있다.

Ⅳ. 실험 및 토론

제안된 알고리즘의 성능을 평가하기 위해서 4개의 화상데이타와 음성시료를 이용한 실험을 Pentium PC 상에서 수행했다. 화상데이타는 각각 512×512 흑백 정지 화상이며 각 화상의 화소는 256개의 흑백 단계로 이루어져 있다. 본 실험에서 벡터의 차원은 4×4=16을 사용했으며, 제안된 알고리즘은 전체 탐색법 및 ENNS 그리고 EENNS 알고리즘과 실행속도 및 평균거리계산 횟수면에서 비교되었다. 각 실험에서 사용된 코드북은 lena화상으로부터 LBG 알고리즘을 이용하여 구성되었으며, 이 코드북을 이용하여 4개의 화상(lena, peppers, jet, and baboon)을 각각 부호화했다. 코드북 구성 시에는 앞에서 언급한 것처럼 제안된 알고리즘을 응용하여 고속 VQ 학습 알고리즘을 구현하여 사용했다.

표 1과 2는 각 화상을 부호화하는 데 필요한 평균거리 계산 횟수와 실행속도를 코드북의 크기를 128부터 1024까지 변화시키면서 구한 결과이다. 표 1과 표 2를 보면 제안된 알고리즘은 전체탐색 알고리즘에 비해 매우 적은 계산량을 필요로 한다는 것을 알 수 있다. 제안된 알고리즘을 특히 EENNS 알고리즘과 비교하면, 거리계산횟수는 약 75.5%, 실행속도는 약 86.5%로 모든 경우에 EENNS 알고리즘보다 뛰어나다는 것을 알 수 있다. 표 1과 2에서 계산량 감소 폭은 각 화상에 따라 조금씩 다른데 이것은 화상의 데이타가 코드워드를 중심으로 집중되어 있는 정도가 어떤가에 따른 것이다. 일반적으로 부호화된 화상의 MSE가 높을수록 계산량의 감소 폭은 떨어지게 된다.

알고리즘을 부분거리탐색법과 결합시키면 보다 많은 계산량의 감소를 얻어낼 수 있다. 표 3은 이 경우에 각 알고리즘의 실행속도를 비교한 결과이다. 표 2와 표 3을 비교해 보면 제안된 알고리즘은 부분거리탐색법과 결합시켰을 때 그렇지 않은 경우에 비해 약 16%의 실행속도 감소가 있음을 알 수 있다. 표 3의 경우에도 표 2에서와 동일하게 제안된 알고리즘이 항상 ENNS와 EENNS 알고리즘보다 뛰어난 성능을 보임을 확인할 수 있다.

음성시료의 경우에는 3명의 화자가 각각 숫자 음을 0에서 9까지 30회 반복 발음한 데이타를 이용해서 실험했다. 음성시료는 8 Bit PCM 파형이며 8 차원 벡터를 이용하였고, 코드북은 시료 1을 사용하여 LBG 알고리즘으로 얻어냈다.

표 3. 부분거리탐색법을 이용한 경우의 실행속도(단위는 초, 괄호 안은 Full Search Algorithm과 비교한 비율)

크기	방법	화상			
		Lena	Peppers	Jet	Baboon
128	Full(PDS)	9.34(58.82)	9.40(59.01)	15.00(93.87)	11.15(70.00)
	ENNS	0.99(6.23)	1.15(7.22)	1.10(6.88)	3.13(19.65)
	EENNS	1.04(6.55)	1.15(7.22)	1.10(6.88)	2.86(17.95)
	Our Method	0.99(6.23)	1.04(6.53)	0.99(6.20)	2.42(15.19)
256	Full(PDS)	18.45(57.12)	18.56(57.46)	29.77(92.02)	21.91(67.85)
	ENNS	1.54(4.77)	1.76(5.45)	1.81(5.60)	5.44(16.85)
	EENNS	1.43(4.43)	1.65(5.11)	1.65(5.10)	4.89(15.14)
	Our Method	1.37(4.24)	1.54(4.77)	1.48(4.57)	4.23(13.10)
512	Full(PDS)	35.70(55.31)	35.98(55.75)	58.38(90.23)	42.35(65.62)
	ENNS	2.31(3.52)	2.85(4.42)	3.02(4.67)	9.94(15.40)
	EENNS	2.14(3.32)	2.58(4.00)	2.47(3.82)	8.57(13.28)
	Our Method	2.03(3.15)	2.42(3.75)	2.25(3.48)	7.47(11.57)
1024	Full(PDS)	84.04(65.05)	84.04(65.06)	127.81(98.77)	97.06(75.20)
	ENNS	4.45(3.44)	5.93(4.60)	5.38(4.16)	20.66(16.01)
	EENNS	3.85(2.98)	4.95(3.83)	3.90(3.01)	15.71(12.17)
	Our Method	3.63(2.81)	4.45(3.44)	3.57(2.76)	13.30(10.30)

표 4. 부분거리탐색법을 이용한 경우의 평균 거리탐색 횟수 및 실행속도(괄호 안, 단위는 초)

크기	방법	음성		
		시료 1	시료 2	시료 3
128	Full(PDS)	128(24.11)	128(23.18)	128(22.19)
	ENNS	22.66(7.86)	21.48(7.25)	22.35(7.25)
	EENNS	13.24(7.91)	12.83(7.31)	12.83(7.25)
	Our Method	11.35(7.64)	10.97(7.03)	10.97(7.03)
256	Full(PDS)	256(45.70)	256(44.60)	256(43.39)
	ENNS	41.57(13.13)	40.25(12.30)	41.65(12.31)
	EENNS	21.03(12.58)	20.71(11.81)	21.65(11.75)
	Our Method	17.84(12.14)	17.54(11.43)	18.32(11.42)
512	Full(PDS)	512(89.36)	512(87.22)	512(85.13)
	ENNS	73.89(21.97)	72.15(20.76)	74.63(20.76)
	EENNS	33.53(20.49)	33.67(19.39)	35.53(19.39)
	Our Method	28.39(19.94)	28.51(18.84)	30.07(18.89)
1024	Full(PDS)	1024(170.66)	1024(167.14)	1024(159.23)
	ENNS	130.50(36.91)	130.00(35.65)	135.43(35.75)
	EENNS	52.25(34.65)	53.62(33.56)	57.24(33.77)
	Our Method	44.08(33.67)	45.23(32.46)	48.21(32.74)

표 4에는 부분거리탐색법과 결합한 경우에 평균거리계산 횟수와 실행속도가 동시에 표시되어 있다. 표 4를 통해서도 제안된 알고리즘의 평균거리계산 횟수가 가장 적고 실행속도 역시 가장 빠름을 알 수 있는데, 화상에 비해서는 그 성능의 향상 폭이 크지 않다는 것을 또한 알 수 있다. 화상의 경우와 비교할 때 음성데이타의 평균거리가 화상데이타의 평균거리 계산횟수보다 더 많은 것은 음성데이타가 화상데이타보다 평균 부근에 덜 분포되어 있다는 것을 보여준다. 제안된 알고리즘을 특히 ENNS와 비교할 때 평균거리계산 횟수가 적어진 것에 비해 실행시간이 그만큼 줄지 않는 것은 벡터의 차원이 낮아서 매 데이터마다 필요한 분산계산이 오버헤드로 크게 작용하

여 실행시간의 상당부분을 잠식하고 있기 때문이다.

V. 결 론

본 논문에서는 입력벡터의 평균과 분산을 이용한 보다 빠른 VQ 부호화 알고리즘을 제안하였다. 제안된 알고리즘은 평균과 분산 그리고 거리 사이에 성립하는 새로운 관계식을 이용하여 보다 많은 코드워드들을 거리계산으로부터 제외함으로써 계산량의 감소를 얻고 있으며 실험을 통하여 이러한 사실을 확인하였다. 전체탐색법에 비해 제안된 알고리즘은 평균과 분산을 저장하기 위한 $2N$ 개의 부가적인 메모리를 요구하고 있으나 동일한 메모리를 요하는 EENNS 알고리즘에 비해서는 아무런 대가없이 보다 나은 계산량의 감소를 보여준다.

참 고 문 헌

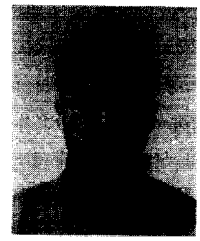
1. R. M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, vol. 1, pp. 4-9, Apr. 1984.
2. N. Moayeri, D. L. Neuhoff, and W. E. Stark, "Fine-coarse vector quantization," *IEEE Trans. Commun.*, vol. COM-33, no. 10, pp. 1132-1133, Oct. 1985.
3. V. Ramasubramanian and K. K. Paliwal, "Fast k-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding," *IEEE Trans. Signal Processing*, vol. 40, no. 3, pp. 518-531, Mar. 1992.
4. C. D. Bei and R. M. Gray, "An improvement of the minimum distortion encoding algorithms for vector quantization and pattern matching," *IEEE Trans. Commun.*, vol. COM-33, no. 10, pp. 1132-1133, Oct. 1985.
5. M. Orchard, "A fast nearest neighbor search algorithm," in *Proc. IEEE ICASSP*, Toronto, Canada, pp. 2297-2300, May 1992.
6. S. W. Ra and J. K. Kim, "A fast mean-distance-ordered partial codebook search algorithm for image vector quantization," *IEEE Trans. Circuits Syst. II*, vol 40, no. 9, pp. 576-579, Sept. 1993.
7. C. H. Lee and L. H. Chen, "Fast closest codeword search algorithm for vector quantisation," *IEE Proc. Vision. Image Signal Process.*, Vol. 141, no. 3, June 1994.

▲백 성 준(Seongjoon Baek) 1967년 1월 16일생

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

▲이 대 룡(Daeryong Lee) 1967년 2월 20일생
1990년 2월 : 한양대학교 전자공학과 졸업(공학사)
1992년 2월 : 서울대학교 전자공학과 졸업(공학석사)
1992년 3월~현재 : 서울대학교 전자공학과 박사과정

▲전 범 기(Bumki Jeon) 1969년 12월 1일생



1992년 2월 : 연세대학교 전기공학과 졸업(공학사)
1995년 2월 : 서울대학교 전자공학과 졸업(공학석사)
1995년 3월~현재 : 서울대학교 전자공학과 박사과정

▲성 굉 모(Koengmo Sung)
현재 : 서울대학교 전기공학부 교수