부영역 기반 코드워드 인덱스 캐시를 사용한 고속 벡터 양자화

(A Fast Vector Quantization using Subregion-based Caches of Codeword Indexes)

김 용 하 † 김 대 진 †† 방 승 양 †††

(Yong-Ha Kim) (Daijin Kim) (Sung-Yang Bang)

요 약 본 논문은 부영역 분할과 코드워드 인덱스의 캐시 개념을 이용하여 벡터 양자화를 위한 고속 코드북 생성 및 부호화 방법을 제안한다. 제안한 방법은 인접한 입력 벡터는 대개 코드북내 특정 코드워드에 의해 나타내어지는 국부성에 바탕을 두고 있다. 초기에 모든 학습 벡터가 거리에 기반한 근접성을 이용하여 정해진 수의 부영역으로 분할된다. 각 부영역에 하나의 코드워드 인덱스 캐시가 할당되는데 이 캐시는 학습 초기에는 전체 코드북 크기에 대응하는 코드워드 인덱스를 갖는다. 학습이 진행되면서 입력 벡터가 갖는 국부성 때문에 각 부영역내 캐시중 사용되지 않는 코드워드 인덱스가 점차 발생하게 되므로 이들은 LRU(Least Recently Used) 삭제 알고리즘에 의해 제거된다. 학습이 진행됨에 따라 부영역 캐시에는 주어진 입력 벡터에 의해 참조되는 코드워드 인덱스만이 남게 되므로 한 학습 주기 동안 필요한 학습 시간이 점차 짧아지게 되어 전체적으로 코드북 생성 시간을 크게 줄일 수 있게 된다. 제안한 방법은 매 학습주기마다, 코드워드 인덱스 삭제 후보 중 주어진 부영역 중심으로부터 거리에 의해 멀리 떨어진 것부터 반만을 제거함에 따라, 복원된 영상의 화질 열화가 거의 없다.

시뮬레이션 결과 제안한 방법은 기존의 LBG 방법에 비해 화질 열화는 거의 없지만 코드북 생성 (또는 부호화) 속도를 2.6-5.4배 (또는 3.7-18.8배) 향상시킨다.

Abstract Abstract A fast codebook generation method using the sub-region based caches of codeword indexes is proposed for vector quantization. The proposed method exploits the localization property that the proximate input vectors are usually represented by a specific part of codewords in the codebook. Initially, all training input vectors are partitined into multiple disjoint subregions of input vectors according to their proximities. A cache of the codeword indexes is assigned to each individual subregion, where the cache has a full size of codeword indexes in the very begining of learning and it maps the input vectors into the corresponding codewords through the codeword indexes. Due to the localization property, non-matched codeword indexes will be occurred and they are purged from the cache because they will not be refereneced any more. As the iteration goes on, a small number of codeword indexes that are referenced by the input vectors in the subregion are remained in the cache. The proposed scheme reduces the codebook generation time greatly because it computes the distortion with only relavant codeword indexes in the corresponding cache. It does not also degrade the recovered image quality too much because the purging policy is conservative and safe by discarding half of the non-matched codeword indexes at every iteration and the furthest codeword index from the prototype of the subregion first. Simulation results show that the proposed method speeds up the codebook generation time (or encoding time) by 2.6-5.4 times (or 3.7-18.8 times), respectively, than the LBG full search method, without sacrificing the recovered image quality.

• 본 연구는 과학재단 특정기초연구(1999-2-30200-047-3)의 연구비 지원에 의해 이루어졌음

† 비 회 원 : (주)판타그램연구실 연구원
ysoya@phantagram.com
†† 정 회 원 : 포항공과대학교 컴퓨터공학과 교수
dkim@postech.ac.kr
††† 종신회원 : 포항공과대학교 컴퓨터공학과 교수
sybang@postech.ac.kr

논문접수 : 2000년 8월 24일
심사완료 : 2001년 3월 8일

1. 서 론

정보를 압축하여 전송율(bit-rate)을 낮추는 방법은 오래전부터 다양한 방식으로 음성이나 영상 데이타에 적용되었다. 최근 들어 컴퓨터의 발달로 방대한 멀티미디어를 저장하고 전달하기 위한 여러 방법들이 제안되고 있는데, 그 중 손실 압축 방법의 대표적인 것으로 벡터 양자화 기법을 들 수 있다.

벡터 양자화(vector quantization)는 정보원 압축 부호화(source coding)기법의 하나로, 다차원 벡터로 나타내어진 M개의 입력 패턴들을 N개의 코드 벡터(code vectors)로 구성된 코드북(codebook)으로 양자화하는 방법이다[1]. 그림 1은 벡터 양자화의 일반적인 과정을 나타낸 것이다. 벡터 양자화는 주어진 왜곡 척도(distortion measure)를 이용하여 입력 벡터 패턴을 가장 근접한 패턴 (코드 벡터)으로 매핑하는 일종의 패턴 인식이라 할 수 있다. 또한, 벡터 양자화는 데이타의 분류와 선형 변환과 같은 여러 종류의 복잡한 신호 처리 분야의 전처리 과정으로 사용될 때는 복잡한 데이타 처리 과정을 간단한 대조표(lookup table)를 사용하여 처리하므로 계산 복잡도를 크게 낮출 수 있다. 이런 이유로 지난 수 년 동안 벡터 양자화는 음성 인식이나 음성, 영상 등의 멀티미디어 압축 분야에서 매우 중요한 기술로 응용되고 있다.

그러나, 벡터 양자화에 필요한 계산량은 벡터의 차원 및, 코드북의 크기의 지수 함수에 비례하여 증가한다[1]. 근래에 있어서는 반도체 기술의 발달로 메모리 문제는 어느 정도 해결되어 가고 있기 때문에 코드북의 크기는 큰 문제가 아니고, 주로 양자화기의 성능 향상과 계산량의 감축에 큰 비중을 두어 연구가 진행되고 있는 추세이다[2].

전체 왜곡오차(distortion error)를 최소화 하면서 코드북 생성에 소요되는 시간을 감소하기 위한 여러 가지 고속 벡터 양자화 방법이[9]-[14] 제안되었다. 본 논문은 코드북 생성 및 부호화 하는데 걸리는 연산시간을 단축시키고자 입력 벡터의 국부성에 기반한 캐시(cache)를 이용하는 고속 벡터 양자화 방법을 제안한다. 제안한 방법은 입력 벡터의 국부성을 이용하여 인접하는 입력 벡터를 부영역으로 분류하고, 부영역에 있는 입력 벡터들을 독립적으로 취급하여 각 부영역에 대응하는 코드 벡터들을 학습을 통해 얻어, 해당 코드 벡터들의 인덱스만을 캐시에 저장한다. 이에 의해 입력 벡터는 그 입력 벡터가 속하는 부영역의 캐시만을 탐색하면 되므로, 코드북 탐색 시간을 크게 단축할 수 있다.

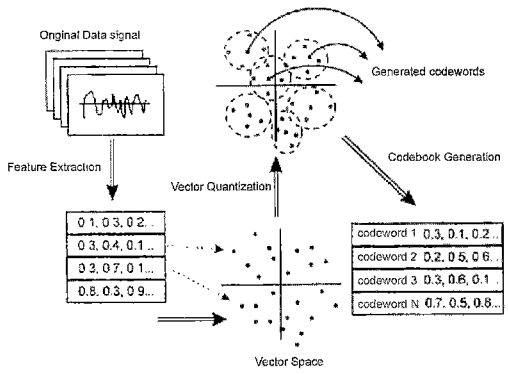


그림 1 벡터 양자화 과정

본 논문의 구성은 다음과 같다. 2장에서는 기존에 주로 사용되던 벡터 양자화 방법을 살펴보고, 3장에서는 제안한 고속 벡터 양자화 방법을 설명한다. 4장에서는 제안한 방법의 유용성을 알아보기 위한 여러 실험 결과를 설명하고 이를 분석한다. 마지막은 결론으로 맺는다.

2. 기존의 벡터 양자화 방법

이 장에서는 일반적인 벡터 양자화기의 구조와 개념, 코드북의 설계 방법 및 탐색 기법들을 기술하고 그 문제점에 대해 논한다.

2.1 벡터 양자화기

벡터 양자화는 입력 신호를 적당한 크기의 k차원 벡터로 규격화한 뒤, 정의된 왜곡 척도(distortion measure)에 의해 코드북(codebook)으로 부터 왜곡이 최소인 코드 벡터(code vector)를 찾아 그것의 인덱스(index)를 전송하여 전송율(bit-rate)을 줄이는 정보원 압축 부호화(source coding)기법의 하나이다[3]. 벡터의 차원 k를 무한히 크게 하면 벡터 양자화기는 이론적으로 rate-distortion 한계에 근접하는 성능을 보인다. 즉, 주어진 평균 왜곡 D에 대하여 벡터 양자화기는 가장 낮은 전송율로 정보를 압축할 수 있고, 평균 전송율 R이 주어졌을 때 가장 낮은 왜곡으로 정보를 압축할 수 있다. 한 벡터가 가지는 왜곡 척도로는 제곱 오차라 불리는 정의식을 사용하는데, 한 벡터 X와 i번째 부호벡터 Ci와의 제곱 오차는 식 (1)로 정의된다.

$$d(X, C_i) = \sum_{j=0}^{k} (X_j - C_{ij})^2 \qquad (1)$$

여기서 k는 입력 벡터의 차원을 나타낸다.

왜곡 척도로 제곱오차를 사용하여 한 입력 벡터에 대해 최소 거리를 갖는 코드북 내의 한 코드 벡터를 찾는

다면, 전역 탐색(full search)을 통해 찾는 경우, 벡터 양자화를 수행하는데 곱셈 연산 : $k \cdot N = k \cdot 2^{Rk}$, 덧셈(뺄셈 포함) 연산 : $(2k-1) \cdot N = (2k-1) \cdot 1^{Rk}$, 비교 연산 : $N-1 = 2^{Rk}-1$이 요구된다. 일반적인 벡터 양자화기는 부호기와 복호기로 구성되는데 부호기의 역할은 입력벡터 X가 코드북내 k차원 N개의 코드 벡터 중 가장 가까운 코드 벡터를 탐색하여 알아내는 것이고, 복호기는 전송된 index를 이용, 코드북내 코드 벡터를 table look-up 과정을 통하여 재생 벡터 $\hat{X}$로 복원한다.

2.2 LBG(Linde-Buzo-Gray) 알고리듬

코드북 생성 알고리듬 중에서도 대표적인 것이 1980년에 Linde, Buzo, Gray에 의해 개발된 LBG 알고리듬이다[4]. 이들은 Lloyd의 스칼라 양자기의 설계 방법으로 제안한 학습 방법[1]을 일반화하였다. 즉, 주어진 벡터 소스의 통계적 모델이나 학습 데이타를 가지고 벡터 양자화기의 코드북을 설계하는 효율적인 방법을 제안하였다. 이 알고리듬은 선택한 입력 벡터 집합을 N개의 부공간(N은 코드북의 크기)으로 분할한 후, 각 부공간의 중심과 새로운 평균 왜곡을 계산하고, 평균 왜곡이 주어진 문턱값(threshold)보다 작을 때까지 부공간 분할 및 평균 왜곡 계산을 반복해서 수행한다. 그리고 이 알고리듬에서 사용하는 평균 왜곡은 단조 감소하므로 반복하는 동안 더 높은 평균 왜곡을 가지는 코드북이 될 수 없다. 따라서 이 알고리듬은 한정된 입력 벡터 집합에 대해 항상 한정된 학습 과정 내에 평균 왜곡이 수렴하여 벡터 양자기에 사용되는 코드북을 만들 수 있다[1].

VQ를 위한 코드북 설계에 있어서 LBG [4]와 같이 반복적인 수행을 필요로 하는 알고리듬은 초기 코드북을 필요로 한다. 이러한 초기 코드북이 좋으면 좋을수록 최적의 코드북으로의 수렴속도도 빨라지고 성능도 향상되므로, 초기치 생성은 중요한 문제이다[5]. 먼저 무작위 추출 방법(random selection method)[1]은 입력 벡터 집합에서 임의로 서로 다른 N개를 선택하여 초기 코드북으로 사용하는 것인데, 입력 벡터들 사이의 상관계수(correlation)가 높을수록 더 좋은 코드북을 만들 수 있으며, 가장 단순한 접근 방법이다.

제거 방법(pruning method)[6]은 전체 입력 벡터 집합으로부터 시작해서 최종 집합인 코드북이 남을 때까지 입력 벡터를 코드북으로부터 선별적으로 제거해 가는 과정이다. 초기 코드북에 첫번째 입력 벡터를 넣고, 다음 입력 벡터 사이의 왜곡을 계산한다. 이 값이 어떤 정해진 문턱값(threshold)보다 작으면 계속하고, 크면 새로운 코드 벡터로서 코드북에 넣는다. 이후, 각 입력 벡터에 대하여 코드북에 있는 코드 벡터와의 왜곡을 계산하는 위의 과정을 반복하여 코드북에 충분한 코드 벡터가 채워질 때까지 반복한다. 코드북에 들어갈 수 있는 코드 벡터가 원하는 코드 벡터보다 많을 경우는 문턱값을 증가 시킨다.

2.3 고속 벡터 양자화(Fast Vector Quantization)

알고리듬LBG 알고리듬은 왜곡오차를 최소화 하면서 최적의 코드북을 생성하기 위해 모든 코드벡터들을 전역탐색(Full Search)해야 하는 계산부담으로 인해, 화질의 성능면에서는 우수하지만 많은 시간이 소요된다는 단점이 있다. LBG 알고리듬이 널리 알려지기 시작한 초창기에는 주로 어떤 초기 코드북을 선택해야 코드북 생성속도를 줄일 수 있느냐에 대한 연구가 많았으나, 최근에는 코드벡터를 비교하는 횟수를 줄여서 계산량을 줄이거나, 많은 계산시간을 차지하는 유사도 측정에 소요되는 시간을 줄이는 방법이 연구 되어지고 있다.[12] [14] 대표적인 FVQ 알고리듬 두 가지 접근 방법의 예를 들면 다음과 같다.

먼저 Lee와 Chen이 1995년 제안한 평균 피라미드(Mean Pyramid)구조를 이용한 코드북 생성 알고리듬[9]의 경우 이미지(Image)를 계층적인 피라미드(Pyramid)형태로 코딩(coding)하면 보다 상위레벨의 코드벡터와 유사한 벡터일수록 하위레벨에서도 유사할 확률이 높으며, 상위레벨에서 유사도가 적은 코드벡터에서는 하위레벨에서도 유사도가 적다[9]는 성질을 이용하여, 벡터의 차원이 낮은 상위레벨에서 유사도가 가장 큰 코드벡터에 대해서만 하위레벨의 코드벡터와 비교하게 됨으로 인해, 전역 탐색해야 하는 경우에 비해 비교횟수가 큰 폭으로 감소하게 된다. 이로써 코드북 생성시간을 단축하는 접근 방법이다.

PNN(Pairwise nearest neighbor algorithm)[1][15]는 최적의 초기 코드북을 생성하고 초기화 시간을 줄여서 LBG 알고리듬이 빠르게 수렴할 수 있게 하는 접근 방법이다.

고속 벡터 양자화(FVQ)알고리듬의 목표는 LBG 알고리듬을 적용했을 때의 화질수준을 유지하면서도, 코드북 생성 및 부호화 시간을 단축해야 한다는 것이다.[9]

3. 제안한 코드북 벡터 양자화 방법

3.1 착안점

코드북 생성 과정의 관측 결과, 입력 벡터가 소속되는 코드 벡터에는 국부성(localization property)[7]이 있음을 알 수 있다. 여기서 국부성이란 인접한 입력 벡터들이 코드북의 특정 코드 벡터들과 관련되어진다는 것이

다. 이러한 성질을 이용하여 인접하는 입력 벡터를 부영역으로 분류하고, 부영역에 있는 입력 벡터들을 독립적으로 취급하여 각 부영역에 대응하는 코드 벡터들을 학습을 통해 얻어, 해당 코드 벡터들의 인덱스만을 캐시에 저장한다. 이에 의해 입력 벡터는 그 입력 벡터가 속하는 부영역의 캐시만을 탐색하면 되므로, 코드북 탐색 시간을 크게 단축할 수 있다.

일반적인 벡터 양자화기에서 전역 탐색(full search)을 할 경우, 각 입력 벡터에 대한 최소 왜곡을 갖는 코드 벡터 $\hat{x}_m$을 찾기 위해 코드북 전체를 탐색해야 한다. 이것은 전송율에 대해 지수 함수적으로 증가하는 코드북 크기를 고려할 때, 매우 비효율적인 방법이다.

그런데, 코드북 생성 과정에서 어떤 입력 벡터 x_t에 대해 학습 시간 t에서 할당되는 최소 왜곡의 코드 벡터 $\hat{x}_m^{(t)}$를 살펴보면, 그 할당 양태가 불규칙한 것이 아니라 코드북 내의 특정한 작업 집합으로만 할당되고 있음을 알 수 있다. 본 논문에서는 코드북 설계 과정에서 한 입력 벡터가 소속 가능한 코드북 내의 코드 벡터의 인덱스 집합을 작업 집합으로 정의한다. 그런데, 이러한 작업 집합의 크기는 코드북 전체의 크기에 비해 훨씬 작기 때문에, 입력 벡터가 사용하는 작업 집합을 알 수 있으면 탐색에 걸리는 시간을 크게 감소시킬 수 있다.

i번째 입력 벡터 x_i에 대한 작업 집합을 W_i, 전체 코드북에 대한 인덱스의 집합, 전체 코드북을 A라 할 때, 다음과 같은 관계가 성립한다.

$$\hat{x}_i \in \hat{x}_{Wi} (A=\{\hat{x}_i : i=, 2,...N\}, W_i=\{j : j \in 1, 2,...N\}) \quad (2)$$

따라서, 작업집합이 완벽하다면 $i=1, 2, \cdots N$의 모든 $\hat{x}_i$를 탐색할 필요가 없이, Wi에 들어 있는 인덱스(j)에 대해서만($\hat{x}_{Wi}$) 탐색해도 전체 코드북을 검색한 것과 같은 결과를 얻을 수 있을 것이다. 또한, 작업 집합의 크기는 일반적으로 $\|W_i\| << N$의 관계가 성립하므로, 이와 같은 탐색은 전역 탐색보다 훨씬 빠르게 수행될 수 있을 것이다. 즉, 작업 집합은 일종의 캐시(cache)[8]로 작용하므로 본 논문에서는 이를 캐시라고 부른다.

그런데, 모든 입력 벡터에 대해서 캐시를 하나씩 유지하고 계산하는 것은 오히려 계산량을 증가 시킬 뿐 아니라, 최악의 경우 각 입력 벡터 마다 코드 북 크기 N만큼의 인덱스 공간을 차지하게 되므로, 저장 공간도 커지게 된다. 또한, 새로 들어오는 입력 벡터의 경우는 참조하고자 하는 캐시가 존재하지 않기 때문에, 새로이 해당 벡터에 대한 캐시를 생성시켜야만 한다. 그리고, 코드북을 전역 탐색하여 얻은 코드 벡터와 캐시에 있는 인덱스로만 탐색하여 얻은 코드 벡터가 다르게 되는 경우가 일어날 수 있다. 이러한 경우, 캐시에서 얻어진 불완전한 코드 벡터는 그대로 양자화 에러가 되어 양자화기의 성능을 저하시키게 된다.

따라서, 코드북 생성 시에 입력 벡터를 먼저 몇 개의 부영역으로 분할한 뒤, 각 부영역에 대해서 캐시를 할당하는 방법을 사용하고자 한다. 이렇게 하면, 캐시를 유지하기 위해 필요한 메모리 크기와 계산량을 최소화할 수 있을 뿐 아니라, 새로운 입력 벡터에 대해서는 부영역을 참조하여 캐시를 얻어 내면 되므로 부호기로서도 사용할 수 있다. 마지막으로, 같은 부영역에 속한 입력 벡터들은 상관관계가 크고, 각 입력 벡터들에 대응되는 코드 벡터들도 상관관계가 크므로, 결국 각각의 입력 벡터에 대해 따로 캐시를 마련하는 것에 비해 부영역에 대한 캐시를 마련하는 것이 코드북의 불일치성을 최소화 하는 데에 효과적이다.

3.2 제안한 방법

본 절에서는 캐시를 이용하는 코드북 생성 방법과, 구성된 캐시를 이용하여 부호화(encoding)하는 방법을 설명한다. 복호화(decoding) 방법은 기존과 동일하다.

코드북의 생성은 크게 두 단계로 나누어 지는데, 첫번째 단계에서는 입력 벡터 집합을 부영역으로 분할하는 것이고, 두 번째 단계에서는 최종 코드북을 생성하며 부영역에 대한 캐시를 구성하는 것이다.

부영역의 분류 과정은 앞 장에서 기술한 일반적인 양자화 과정과 동일하다. 부영역 크기 $L(<<N)$ 만큼의 부영역 코드북 $A'=\{\hat{x}'_k : k=1, 2,..., L\}$을 생성하고, 결과적으로 각 입력 벡터 x_t에 대하여 부영역에 대한 인덱스 k를 얻게 된다.

다음 단계에서는 캐시 탐색을 통한 코드북 생성을 하게 되는데, 시작에 앞서 부영역에 대해서 캐시 W_k을 할당하며, 각 캐시는 처음에 크기 N의 최종 코드북에 대한 인덱스를 모두 갖는다. 캐시 안에서의 코드 벡터 탐색 및 코드 벡터 갱신은 2장에서 기술한 기존의 일반적인 양자화 과정과 동일하다. 이 단계가 끝나면 결과적으로 하나의 최종 코드북 $A=\{\hat{x}_j : j=1, 2,..., N\}$과, 각 입력 벡터 x_t에 대하여 코드북에 대한 인덱스 j를 얻게 되는데, 각 캐시는 학습 횟수 t가 증가함에 따라, 사용되지 않는 인덱스를 버리므로 학습이 진행됨에 따라 캐시의 크기는 다음과 같이 변화한다.

$$\begin{aligned} W_{k,1} &= \{1, 2,...N\}, \ \|W_{k,1}\| = N \\ W_{k,2} &= \{j \mid j \in 1, 2...N\}, \ \|W_{k,2}\| \le \|W_{k,1}\| \\ W_{k,3} &= \{j \mid j \in 1, 2...N\}, \ \|W_{k,3}\| \le \|W_{k,2}\| \end{aligned} \quad (3)$$

그림 2는 제안한 코드북 생성 방법의 수행 과정을 나타내고 있으며, 그 중에서 캐시 관리 과정을 보다 자세히 설명하면 다음과 같다. 각 학습 과정 t에서 입력 벡터에

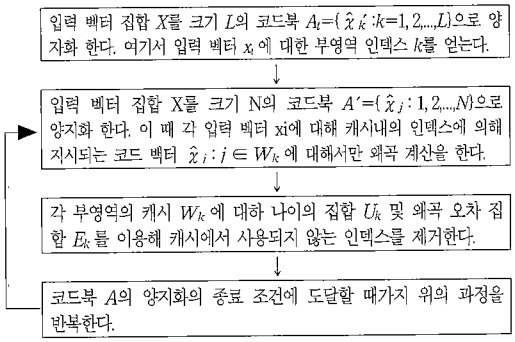


그림 2 제안한 코드북 생성 알고리즘

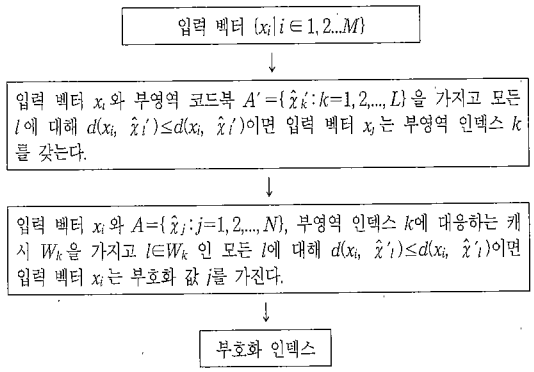


그림 3 제안한 부호화 알고리즘

에 대해 코드북 A의 코드 벡터를 탐색할 때, 해당 입력 벡터가 속한 부영역 k의 캐시 W_k를 참조하여, 캐시 안에 들어 있는 인덱스의 코드 벡터만을 탐색한다. 그런데, 캐시 내 인덱스는 고유한 나이 $u_j \in U_k$를 가지고 있으며, 모든 인덱스의 나이는 매 학습 횟수 마다 하나씩 증가한다. 나이는 해당 인덱스의 코드 벡터가 선택될 경우 즉, 부영역 k에 속한 입력 벡터 x_i에 대해서, W_k의 인덱스 중 $d(x_i, \hat{x}_j) = \min_{j \in W_k} d(x_i, \hat{x}_j)$를 만족하는 인덱스가 j일 때, 해당 인덱스 j의 나이 u_j를 0으로 초기화한다.

하나의 학습이 끝나면, u_j가 나이의 임계치 D보다 크거나 같은 인덱스 j들을 W_k로부터 제거 시켜야 하는데 논 논문에서는 해당 인덱스가 가리키는 코드 벡터와 각 부영역에 포함된 입력벡터 사이의 왜곡의 합을 오름차순으로 정열한 뒤 하위부터 제거하는 방법을 택하였는데 그 방법을 자세히 설명하면 다음과 같다.

위에서 설명한 입력 벡터에 대한 왜곡 계산 과정에서, 각 입력 벡터 x_i와 코드 벡터 $\hat{x}_j$ 사이에 왜곡치 $d(x_i, \hat{x}_j)$가 계산될 때마다 그 값을 $e_j \in E_k$에 더해 나간다. 즉 $e_j = \sum_{x_i} d(x_i, \hat{x}_j)$. e_j 값이 큰 코드 벡터일수록 부영역 k와의 상관관계가 적다고 간주할 수 있다. 따라서, 임계치에 이른 인덱스 중 e_j가 큰 것부터 캐시 W_k로부터 제거된다.

코드북의 부호화는 기본적으로 일반적인 양자화 방법을 따르되, 앞서 설명한 코드북 생성 과정에서 만들어진 부영역 코드북 $A'=\{\hat{x}_k' : k=1,2,...,L\}$과 캐시 $W_k=\{j | j \in 1,2...N\}$를 참조한다.

그림 3은 제안한 부호화 방법의 수행 과정을 나타내었는데 그 중, 캐시 관리를 보다 자세히 설명하면 다음과 같다. 첫번째 단계로, 입력 벡터 x_i와 부영역 코드북 A'의 코드 벡터들 사이의 왜곡을 계산하여 가장 왜곡이 작은 $d(x_i, \hat{x}'_k) = \min_{\hat{x}'_k \in A'} d(x_i, \hat{x}'_k)$, 코드 벡터 $\hat{x}'_k$를 찾는다. 그 인덱스 k를 입력 벡터의 부영역 인덱스로 간주한다. 두 번째 단계로, 입력 벡터 x_i와 그 부영역 캐시 W_k의 인덱스가 가리키는 최종 코드북 A의 코드 벡터들 $\{\hat{x}_j | \hat{x}_j \in A, j \in W_k\}$과의 왜곡을 계산하여 그 중 가장 왜곡이 작은 코드 벡터 $\hat{x}_j$를 찾는다. 그 인덱스 j를 입력 벡터 x_i의 부호화 값으로 간주한다.

3.3 계산량 비교

제안한 코드북 생성 알고리즘의 계산량은 다음과 같다. 먼저 부영역 분할의 경우, 입력 벡터에 대한 계산량 곱셈 연산 : $k \cdot L$, 덧셈(뺄셈 포함) 연산 : $(2k-1) \cdot L$, 비교 연산 : $L-1$ 이 필요하다. 여기서 k는 데이타 차원 수, L은 부영역 분할 개수이다. 두 번째 단계인 최종 코드북 생성시의 계산량은 곱셈 연산 : $k \cdot \| \overline{w} \|$, 덧셈(뺄셈 포함) 연산 : $(2k-1) \cdot \| \overline{w} \|$, 비교 연산 : $\| \overline{w} \| - 1$이며 k는 데이타 차원 수, $\| \overline{w} \|$는 평균 캐시 크기이다. 그리고, 매 학습 과정이 끝난 시점에서 일괄적으로 이루어지는 캐시 관리를 위해서 비교 연산 $\| \overline{w} \| \cdot L$이 추가로 필요하다. 만약 앞서 설명한 캐시에서의 인덱스를 제거하는데 요구되는 계산량은 $\| \overline{w} \| (1+\log \| \overline{w} \|) \cdot L$이 된다. 그런데, 코드북 생성에 사용되는 입력 벡터의 수가 코드북의 크기보다 훨씬 큰 것이 일반적이고, (즉, N<<M) 컴퓨터에 있어서 비교 연산을 수행하는 데 걸리는 시간은 곱셈이나 덧셈 연산 보다 상대적으로 매우 짧은 시간 안에 처리되므로, 비교 연산에 소요되는 시간은 거의 무시할 수 있다고 가정하면, 일반적인 알고리듬의 계산량과 제안된 알고리듬의 계산량의 비율은 $N : (L+\| \overline{w} \|)$으로 근사화 된다.

한편, 얻어진 코드북을 이용하여 임의의 입력 벡터를 부호화하는 경우의 계산량을 비교하면 다음과 같다. 먼

저 일반적인 벡터 양자화 알고리듬에서 필요한 계산량은 곱셈 연산 : $K \cdot N$, 덧셈(뺄셈 포함) 연산 : $(k-1) \cdot N$, 비교 연산 : $N-1$이 필요한데 여기서, k는 데이타 차원 수, N은 코드북 크기이다.

또, 제안된 알고리듬을 이용하여 부호화 할 때 필요한 계산량은 곱셈 연산 : $k \cdot (L+\| \overline{w} \|)$, 덧셈(뺄셈 포함) 연산 : $(k-1) \cdot (L+\| \overline{w} \|)$, 비교 연산 : $(L+\| \overline{w} \|)-1$이 요구되며, 여기서 k는 차원 수, L은 부영역 분할 개수, $\| \overline{w} \|$는 평균 캐시 크기이다. 따라서 부호화시 일반적인 알고리듬의 계산량과 제안된 알고리듬의 계산량과의 비는 $N:(L+\| \overline{w} \|)$가 된다. 그런데, 제4장에서의 실험 결과에 의하면, 아래 식과 같이 부영역 분할 개수 L과 코드북 생성 뒤의 평균 캐시 크기 $\| \overline{w} \|$의 곱은 항상 코드북 크기 N에 비례하는 것을 알 수 있다.

$$N = c \cdot (L \cdot \| \overline{w} \|) (\text{c는 임의의 상수}) \quad (4)$$

따라서 부영역 분할 개수를 코드북의 크기의 제곱근에 비례하는 상수로 설정하면, 코드북 생성 또는 입력 벡터의 부호화 시 일반적인 알고리듬과 제안된 알고리듬의 계산량의 비는 다음과 같다.

$$N:(L+\| \overline{W} \|) \approx N:(c'\sqrt{N}+c''\sqrt{N}) = N:c\sqrt{N} \quad (5)$$

여기서 c′, c″, c 는 모두 임의의 상수이다. 따라서, 제안된 벡터 양자화 알고리듬은 일반적인 벡터 양자화 알고리듬에 비해 코드북 크기의 제곱근($\sqrt{N}$)에 비례하는 계산량 이득을 얻게 된다.

4. 시뮬레이션 결과 및 분석

4.1 시뮬레이션 방법

이미지 데이타의 압축 테스트에 많이 사용되는 LENA 이미지와 BABOON 이미지를 입력 벡터 집합으로 하였다. 이미지의 해상도는 가로 세로 각각 512 픽셀로 이루어져 있으며, 각 픽셀은 256 단계의 흑백 계조(gray level)를 갖는다. 한 입력 벡터는 이미지의 4 x 4 블록으로부터 얻어진 16차원을 갖는다. 이미지 전체의 크기가 512×512이므로, 전체적으로 얻어지는 입력 벡터의 수는 16384개가 된다. 이에 최종 코드북의 크기를 변화 시켜 보면서 제안한 알고리듬을 이용해 코드북을 생성해보았다. 부영역 분할 개수는 여러 번의 시뮬레이션 결과로부터 최종 코드북 크기의 제곱근에 상수 2를 곱한 것으로 정하였다. 코드북의 갱신에는 LBG 알고리듬을 사용하였으며, 3장에서 제안한 코드워드 인덱스 제거하는 방법을 사용하였다. 공통적으로 사용된 상수는 다음과 같다. 입력 벡터의 수 : 16384, 입력 벡터 및 코드 벡터의 차원 : 16, 인덱스의 나이의 임계치 : 3, 임계치에 이른 인덱스 제거 비율 : 65%, LBG 알고리듬의 종료 조건 : 학습 횟수 25이다.

표 1은 LENA 이미지와 BABOON 이미지에 대해 일반적인 LBG 벡터 양자화 알고리듬과 제안한 벡터 양자화 알고리듬을 사용하여 코드북 생성을 수행하는데 걸린 수행 시간과 PSNR을 비교한 것이다. 일반적인 벡터 양자화 방법에 비해 PSNR의 감쇄가 거의 없으면서도 3배 ~ 6배 빠른 속도를 내고 있는 것을 볼 수 있다.

특히, 코드북의 크기가 증가할수록 속도 향상도 커지고 있는 것을 관찰할 수 있다. 제안한 벡터 양자화 알고리듬의 코드북 생성에서는, 초기에 부영역 캐시에 모든 인덱스가 다 있으므로, LBG와 동일한 계산량을 보이다가, 반복이 지나면서 인덱스가 제거됨에 따라 속도 향상을 얻게 되는데, (식 3.2) 반복 수가 더 많아질수록 이 속도 향상의 영향이 크게 나타난다.

특히 LBG 알고리듬의 종료를 위한 문턱값 ε을 좀 더 엄격하게 설정할 경우나, SOFM과 같은 적응형 알고리듬을 사용하는 경우는 반복 횟수가 훨씬 커지므로, 큰 속도 향상을 얻을 수 있다.

또, 위 실험에서는 수명의 임계치와 임계치의 인덱스 제거 비율을 고정시켰지만 실제 적용에서는 코드북의 크기가 큰 경우, 수명의 임계치를 낮추고 제거 비율을 보다 크게 하여 코드북 생성의 속도 향상을 얻을 수 있다.

표 1 Lena 및 Baboon 이미지의 벡터 양자화 결과

Image	Codebook Size	Algorithm	Generation Time(sec)	PSNR
Lena	256	Original LBG	1441	31.6596
		Proposed	471	31.5770
	512	Original LBG	3025	32.5464
		Proposed	820	32.4638
	1024	Original LBG	6050	33.3701
		Proposed	1389	33.3413
	4096	Original LBG	24197	35.8875
		Proposed	4414	35.8791
Baboon	256	Original LBG	1263	24.1412
		Proposed	482	24.1258
	512	Original LBG	2526	24.8472
		Proposed	764	24.8132
	1024	Original LBG	5053	25.6288
		Proposed	1245	25.5936
	4096	Original LBG	20213	27.9803
		Proposed	3743	27.9728

Original Lena Image

Error images

Original LBG

Proposed Algorithm

N=256

N=1024

N=4096

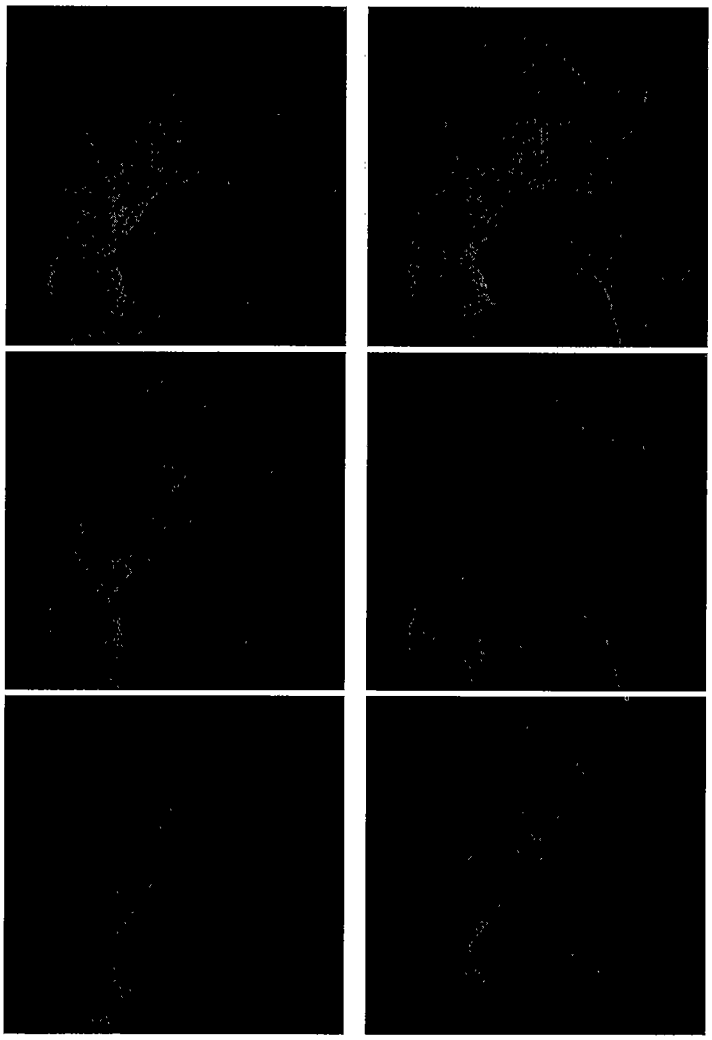
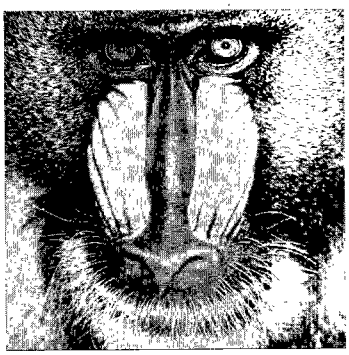


그림 4 코드북 생성 뒤의 Lena 이미지의 화질 비교



Original Lena Image

Error images

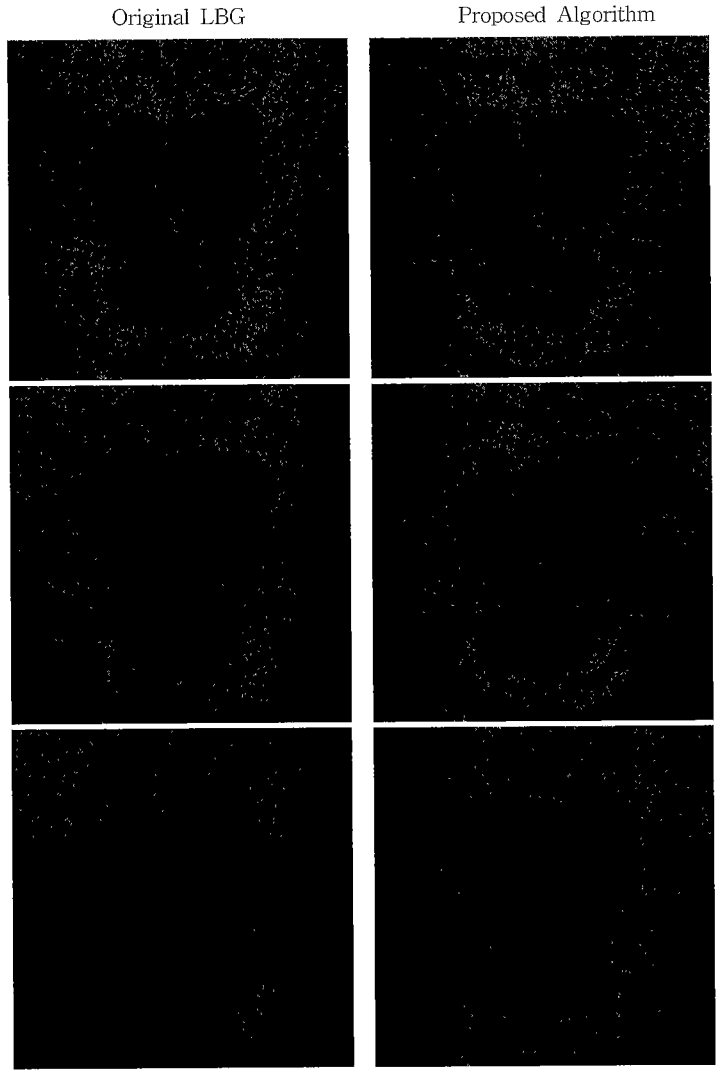


그림 5 코드북 생성 뒤의 Baboon 이미지의 화질 비교

그림 4와 5는 각각 Lena와 Baboon 이미지에 대해 얻어진 코드북을 이용하여 복원한 이미지와 원래 이미지의 차를 나타낸 것이다. 코드북 크기가 커질수록 차가 적어지는 것을 볼 수 있으며, 일반적인 벡터 양자화 알고리듬과 제안한 벡터 양자화 알고리듬 사이의 성능 차는 거의 보이지 않는다. 표 2는 제안한 방법에서 코드북 생성에 소요된 반복 수와 최종 캐시 크기의 평균을 보인 것이다.

일반적으로 벡터 양자화기에서는 코드북 생성에 소요되는 시간보다는, 코드북 생성이 완료된 뒤, 입력 벡터의 부호화(encoding)에 소요되는 시간이 더욱 중요하다.

표 2 제안한 방법에 의한 코드북 생성시의 입출력 파라메터

Image	Codebook Size	Number of sub-regions	Number of iterations for partitioning of sub-region	Number of iterations for codebook generation	average size of final cache $\|\overline{w}\|$
Lena	256	32	158	20	18.15
	512	46	18	16	23.72
	1024	64	15	13	30.58
	4096	128	19	8	67.19
Baboon	256	32	28	19	34.51
	512	46	20	15	41.79
	1024	64	21	13	50.69
	4096	128	20	8	80.47

표 3 10000개의 입력 벡터를 부호화 하는 데 걸리는 시간

Image	Codebook Size	TLBG (Encoding time by LBG)	Tproposed (Encoding time by proposed method)	Speedup $=\frac{T_{LBG}}{T_{proposed}}$
Lena	256	21.3	4.4	×4.84
	512	43.2	6.2	×6.97
	1024	83.9	8.1	×10.36
	4096	342.1	17.7	×19.33
Baboon	256	21.5	5.8	×3.71
	512	42.2	7.5	×5.63
	1024	84.6	9.9	×8.55
	4096	345.1	18.3	×18.85

표 3는 생성된 코드북을 이용하여 임의의 입력 벡터 10000 개를 부호화하는 데 걸리는 시간을 일반적인 벡터 양자화 알고리듬(LBG)과 제안한 벡터 양자화 알고리듬에 적용하여 얻은 결과를 나타낸 것이며, 그림 6은 이를 코드북 크기와 시간을 축으로 하여 도시한 것이다.

그림 6을 살펴보면, 기존 알고리듬의 경우 코드북의 크기가 지수적으로 증가함에 따라 계산에 걸리는 시간도 지수적으로 증가하는 것을 볼 수 있다. 이에 비해, 제안된 알고리듬은 코드북의 크기가 지수적으로 증가하더라도, 계산에 걸리는 시간은 선형적으로 증가하는 것을 볼 수 있는데, 이는 식 (5)에서 얻어진 결과와 동일한 것이다.

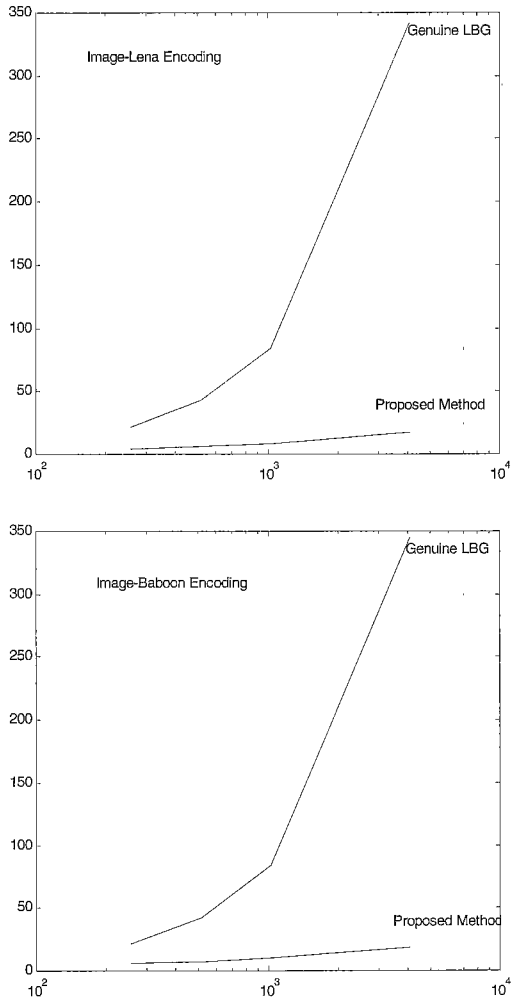


그림 6 부호화에 걸리는 시간 비교

5. 결 론

멀티미디어 데이타의 압축을 위해 이용되는 벡터 양자화 방법은, 입력 벡터와 코드북 사이의 왜곡 척도 연산과 탐색을 반복적으로 수행하여 코드북 생성 및 부호화를 행하는데, 이에 걸리는 시간은 입력 벡터의 개수와 코드북의 크기에 비례하여 증가하게 된다.

본 논문에서는 코드북 생성 및 부호화에 소요되는 시간을 감축하기 위해, 코드북에 대한 기존의 전역 탐색(full search)방법 대신 부영역 분할 및 캐시 탐색을 행하는 효과적인 방법을 제안하였다. 실험 결과, 제안한 새로운 방법은 기존의 방법에 비해 왜곡 오차를 거의 증가 시키지 않으면서도 훨씬 짧은 시간 안에 코드북 생성과 부호화를 행하는 것이 가능했다. 특히 부호화의 경우, 기존의 벡터 양자화 방법의 경우 전송률이 증가함에 따라 부호화 소요 시간이 지수적으로 증가하는 것에 비해, 제안한 방법에 의하면 전송률이 증가하더라도 부호화 소요 시간은 선형적으로 증가하게 된다. 따라서, 본 논문이 제안한 알고리듬은 현재 그 수요가 점차 증가하고 있는 대용량의 멀티미디어 데이타 압축 분야에서 부호화에 걸리는 계산량을 혁신적으로 개선할 수 있는 새로운 방법이 될 수 있을 것으로 기대된다.

향후 연구는 코드 워드의 평균 피라미드를 사용한 고속 벡터 양자화 방법과 제안한 방법을 결합하여 탐색 공간을 줄이면서 벡터간 거리 계산시 연산 복잡도를 줄임으로서 보다 고속화된 벡터 알고리즘을 개발하고자 한다.

참 고 문 헌

[1] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer Academic Publishers, 1991.

[2] R. M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, vol. 1, pp.4-29, 1984.

[3] N.M. Nasrabadi, and R.A. King, "Image Coding using Vector Quantization : A Review," *IEEE Trans. Comm.*, pp.957-971, 1988.

[4] Linde, Buzo, and Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp.84, 1980.

[5] Majid Rabbani, and Paul W. Jones, *Digital Image Compression Techniques*, SPIE PRESS, vol. TT7, 1991.

[6] J.D.M. Auliffe, L.E. Atlas, and C.Rivera, "A Comparison of the LBG Algorithm and Kohonen Neural Network Paradigm for Image Vector Quantization," in Proc. *IEEE. Conf. ASSP*, 1990, pp.2293-2296.

[7] D. Patterson and J. Hennessy, *Computer Organization & Design : The Hardware/ Software Interface*, Morgan Kaufmann Publi- shers, Inc., 1998.

[8] A. Smith, "Cache Memories," *Computing Surveys*, vol. 13, no. 3, pp.473-530, 1982.

[9] Chang-Hsing Lee and Ling-Hwei Chen, " A Fast Algorithm for Vector Quantization Using Mean Pyramids of Codewords," *IEEE Trans. Comm.* Vol. 43, no. 2, pp.1697-1702, 1995

[10] X. Wu and L. Guan, "Acceleration of the LBG Algorithm," *IEEE Trans. Comm.*, vol. 42, pp.1518-1523, 1994.

[11] S. C. Tai, C. C. Lai, and Y. C. Lin, "Two fast nearest neighbor searching algorithms for image vector quantization," *IEEE Trans. Comm*, vol. 44, pp.1263-1268, 1996

[12] Yih-chuan Lin and shen-Chuan Tai, "A Fast Linde-Buzo-Gray Algorithm in Image Vector Quantization," *IEEE Trans. Circuit & systems -II Analog & Digital Signal Processing*, vol. 45, no. 3, 1998

[13] J. H. Li, and N. Ling, "A Novel VQ Codebook Design Technique," *IEEE Trans. On Consummer Electronics*, vol. 43, no. 4, 1997

[14] Chin-Chen Chang and Yu-Chen Hu, "A Fast LBG Codebook Trainning Algorithm for Vector Quantization," *IEEE Trans. On Consummer Electronics*, vol. 44, no. 4, pp.1201-1208, 1998

[15] W. H. Equitz, "A new vector quantization clustering algorithm," IEEE Trans. Acoustic, Speech, Signal Processing, vol. 37, pp.1568- 1575, 1989

김 용 하

1998년 2월 포항공과대학교 컴퓨터 공학과 졸업. 2000년 8월 포항공과대학교 대학원 컴퓨터 공학과 PAMI연구실, 석사학위 취득. 2001년 3월 ~ 현재 (주) 판타그램 연구실. 주관심분야는 영상처리, 패턴인식

김 대 진

1981년 2월 연세대학교 전자공학과(학사). 1984년 2월 KAIST 전기 및 전자공학과(석사). 1991년 8월 Syracuse University, Electrical and Computer Eng., (박사). 1984년 3월 ~ 1986년 12월 한국방송공사 기술연구소. 1992년 3월~1999년 6월 동아대학교 컴퓨터공학과 부교수. 1999년 7월 ~ 현재 포항공과대학교 컴퓨터공학과 부교수. 주관심분야는 지능시스템, 멀티미디어 처리

방 승 양

1966년 일본 Kyoto대학 전기공학에서 학사. 1969년 서울대학교 전기공학에서 석사. 1974년 미국 University of Texas 전산학에서 박사를 받았음. 미국 Wayne State University, NCR, Bell 연구소 등에서 근무하다가 1981년 귀국. 한국전자기술연구소 시스템부 실장, 부장 역임, (주)유니온시스템 전무. 1986년부터 포항공대 컴퓨터공학과 교수. 현재 뇌연구센터 소장. 관심분야는 패턴인식, 신경회로망.