DFS를 이용한 추가 메모리를 요구하지 않는 제로트리 압축기법

김 충 길[†] · 이 주 경[††] · 정 기 동[†††]

요 약

SPIHT는 수행속도가 빠르고 효율적인 웨이블릿 기반의 이미지 압축 알고리즘으로 잘 알려져 있다. 그러나, SPIHT는 알고리즘 수행에 필요한 제로트리 및 계수의 상태를 저장하기 위하여 리스트 구조를 사용하고 있어 추가 메모리를 요구하며, 비트율의 증가에 따라 메모리 요구량이 증가하는 단점을 가진다. 본 논문에서는 SPIHT 알고리즘을 수행하는데 있어 추가 메모리를 요구하지 않는 MZP-DFS 알고리즘을 제안한다. 제안된 기법은 깊이우선 순서에 따라 공간트리를 탐색하고 테스트 함수 및 복원 계수의 LSB를 이용함으로써 추가 메모리를 제거하였으며 SPIHT와 동일한 성능을 가진다. MZP-DFS는 추가 메모리를 요구하지 않기 때문에 하드웨어 제작비용을 절감할 수 있으며, 각각의 공간트리를 병렬적으로 수행할 수 있기 때문에 실시간 이미지 압축에 적합하다.

Zero-tree packetization without additional memory using DFS

Chung-Kil Kim[†] · Joo-Kyong Lee[††] · Ki-Dong Chung[†††]

ABSTRACT

SPIHT algorithm is a wavelet based fast and effective technique for image compression. It uses a list structure to store status information which is generated during set-partitioning of zero-tree. Usually, this requires lots of additional memory depending on how high the bit-rate is. Therefore, in this paper, we propose a new technique called MZP-DFS, which needs no additional memory when running SPIHT algorithm. It traverses a spatial-tree according to DFS and eliminates additional memory as it uses test-functions for encoding and LSB bits of coefficients for decoding respectively. This method yields nearly the same performance as SPIHT. This may be desirable in hardware implementation because no additional memory is required. Moreover, it exploits parallelism to process each spatial-tree that it can be applied well in real-time image compression.

키워드 : Image Compression, SPIHT, Zero-tree

1. 서 론

JPEG 등과 같은 기존의 표준은 8×8 블록에 대한 DCT (Discrete Cosine Transform) 변환을 기반으로 하고 있어 낮은 비트율에서 블록의 경계가 두드러지는 블록킹 현상(Block Effect)이 단점으로 지적되고 있다. 이러한 DCT 변환의 단점을 보완하기 위하여 최근 웨이블릿변환(Wavelet Transform)을 이용한 이미지 압축 기법에 대한 연구가 진행되었으며, EZW(Embedded Zerotree Wavelet)[1], SPIHT (Set Partitioning in Hierarchical Trees)[2]와 같은 알고리즘은 JPEG보다 우수한 성능을 보여주고 있다.

Shapiro는 [1]에서 피라미드 분해된 웨이블릿 이미지의 공간적 자기유사성(spatial self-similarity)를 이용하여 제로트리를 정의하고 EZW 알고리즘을 제안하였다. SPIHT 알고리즘은 새로운 set partitioning 규칙을 제안함으로써 EZW보다 효율적인 비트스트림을 생성한다. 이들 기법들은 알고리즘의 수행과정에서 계수 및 트리의 상태를 유지하기 위하여 리스트 구조를 사용하고 있다. 따라서 리스트의 추가, 삭제 및 이동 등의 논리를 갖추어야 하며, 리스트를 저장하기 위한 메모리 공간을 필요로 한다. 메모리의 요구량은 비트율과 이미지에 의존적이고 가변적이며 비트율의 증가에 따라 요구량도 증가한다. 이러한 점은 이들 알고리즘을 하드웨어로 구현하는데 있어 문제점으로 작용하고 있다.

그 동안 SPIHT의 리스트 사용에 따른 문제점을 해결하기 위한 연구가 진행되었다. [3]에서 Wheeler는 웨이블릿 이미지를 4개의 동일 크기의 공간블럭(spatial block)으로 분할함으로써 리스트의 최대 요구량을 줄이는 기법을 제안하였으며, [4-6]에서는 일정한 크기의 상태맵을 이용하여 리스트를 제거하는 연구가 진행되었다. 특히 공간트리의 탐색 방식에 있어 [4,5]의 연구는 깊이우선탐색(DFS : Depth First Search)을, [6]의 연구는 너비우선탐색(BFS : Breadth First Search)을 기반으로 하고 있다.

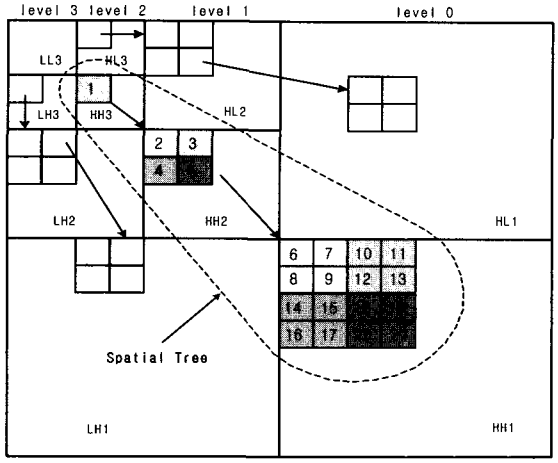
본 논문에서는 SPIHT 알고리즘의 수행에 있어 리스트 및 상태맵과 같은 추가의 메모리를 요구하지 않고 웨이블릿 이미지를 압축할 수 있는 MZP-DFS(Memory-less Zerotree Packetization-DFS) 알고리즘을 제안한다. 본 논문에서는 SPIHT의 리스트와 동일한 기능을 수행할 수 있는 계수 및 트리 테스트기법을 제안하며 복원과정에서 계수의 LSB를 이용한 상태 유지 기법을 제안한다. 본 논문의 구성은 제 2장에서는 관련연구에 대하여 간략히 살펴보고, 제 3장에서 MZP-DFS 알고리즘을 기술한다. 제 4장에서는 실험을 통하여 본

† 준 회 원 : 부산대학교 대학원 멀티미디어협동과정
†† 준 회 원 : 부산대학교 대학원 전자계산학과
††† 종신회원 : 부산대학교 전자계산학과 교수
논문접수 : 2003년 5월 20일, 심사완료 : 2003년 6월 12일

논문에서 제안한 MZP-DFS 알고리즘의 성능을 평가한다. 그리고 제 5장에서는 결론을 맺는다.

2. 관련 연구

Mallat[9]에 의해 MRA(Multi-Resolution Analysis) 기법이 제안된 이후 웨이블릿 이미지를 (그림 2.1)과 같이 트리에 대응시켜 처리하려는 연구가 진행되었으며, EZW와 SPIHT는 이러한 웨이블릿 이미지의 속성을 잘 이용한 압축기법이다.



(그림 2.1) 웨이블릿 이미지의 트리 대응

Shapiro는 [1]에서 임계치(threshold)를 주어 계수(coefficient)의 절대치가 주어진 임계치보다 작은 경우를 비중요계수(insignificant), 반대의 경우를 중요계수(significant)로 정의하였다. 또한 비중요계수로 구성된 공간트리를 제로트리(zerotree)로 정의하였으며, 이를 기반으로 EZW 알고리즘을 제안하였다. EZW는 점진적으로 작아지는 임계치에 대하여 주부호화 과정(dominant pass)과 종속부호화 과정(subordinate pass)을 반복적으로 수행하며, 원하는 조건에서 부호화 과정의 수행을 중지할 수 있다. EZW는 주부호화 과정에서 spatial-tree에 대한 중요계수 테스트(significant test)를 수행하며, 그 결과에 따라 set partitioning 규칙을 적용한다. set partitioning 과정에서 지배적인 에너지를 가지는 계수는 spatial-tree에서 분리된다. 디코더는 전송된 비트스트림으로부터 동일한 set partitioning 규칙을 통하여 공간트리를 재구성하고 계수값을 복원한다.

SPIHT는 더욱 향상된 set partitioning 규칙을 통하여 EZW보다 효율적인 비트스트림을 생성한다. SPIHT는 트리의 구성과 set partitioning 규칙에서 EZW와 차이를 가지는데, EZW에서 트리는 루트를 포함하지만 SPIHT의 트리는 루트 노드를 제외시키며, 루트를 제외한 자손들로 구성된 A type의 descendant tree, 루트와 자식(offspring)을 제외한 B type의 granddescendant tree를 사용하고 있다. set partitioning 규칙에 있어서 SPIHT는 A type 트리가 중요계수를 포함할 때 4개의 자식노드와 1개의 동일한 루트노드를 가지는 B type 트리로 분할되는 과정을 추가로 가진다. B type 트리는 자식을 새로운 루트노드로 하는 4개의 A type 서브트리로 분할된다.

SPIHT는 계수 및 set partitioning 상태를 유지하기 위하여 LIS(List of Insignificant Set), LIP(List of Insignificant Pixel) 및 LSP(List of Significnat Pixel) 리스트를 사용하고 있다. LIS는 제로트리 조건을 만족하는 모든 트리를 원소로 가지며, LIP와 LSP는 트리에서 분리된 비중요계수와 중요계수를 원소로 가진다. SPIHT에서 계수 및 트리에 대한 노드들은 그들이 속한 리스트에 따라 인코딩 동작이 결정된다. SPIHT는 리스트를 저장하기 위해 추가의 메모리를 요구한다. 요구되는 메모리의 크기는 가변적이며, 비트율이 높아질 경우 요구량 또한 증가한다. 이러한 메모리 요구의 특성은 SPIHT 알고리즘의 단점으로 지적되고 있다.

SHPIT의 리스트 사용에 따른 문제점을 해결하려는 연구가 진행되고 있으며, 대표적인 연구로 NLS(No List SPIHT)와 SMAWZ(Significance Map Based Adaptive Wavelet Zerotree)가 있다. NLS는 선형 인덱싱(linear-indexing) 주소기법, 공간트리에 대한 너비우선탐색(Breadth First Search)과 픽셀당 4비트의 상태맵을 사용하고 있다. SMAWZ는 이미지의 픽셀 수가 i개 라고 할 때, $i$bit, $i/4$bit, $i/16$bit 용량을 가지는 3개의 비트맵과 spatial-tree에 대한 깊이우선탐색(Depth First Search)을 사용하고 있다. 이들 알고리즘의 상태맵 크기는 비트율과 무관하며 고정된 크기를 가진다. 그러나 상태맵은 픽셀수에 의존적이기 때문에 이미지가 커질 경우 상태맵을 위한 메모리 요구량 또한 증가하는 문제점을 가진다.

3. MZP-DFS 알고리즘

본 논문에서는 리스트를 사용하지 않고 계수에 대한 인코딩 동작을 결정할 수 있는 테스트 기법을 제안한다. 제안된 테스트 기법은 추가의 메모리를 요구하지 않으며, 단지 중요계수 테스트 함수만을 사용하고 있다. 본 논문에서 사용될 계수 및 트리에 대한 중요계수 테스트 $S_n(T)$는 임계치를 $2^n(n \geq 0)$, 계수값을 $c_{i,j}$, T를 계수집합이라고 할 때

$$S_n(T) = \begin{cases} 1, & \max_{(i,j)\in T}\{|c_{i,j}|\} \geq 2 \\ 0, & otherwise \end{cases} \quad (3.1)$$

식 (3.1)과 같이 정의되며, 하나의 계수에 대한 $S_n(\{(i,j)\})$를 $S_n(i,j)$로 표기한다.

SPIHT 알고리즘에서 리스트에 속한 노드들은 다음의 특성을 만족한다. 임계치가 2^n일 때, LIP에 속한 노드 (i, j)는 모든 $m(m > n)$에 대하여 $S_m(i,j) = 0$을 만족한다. 또한 LSP에 속한 노드 (i,j)는 $S_m(i,j) = 1$을 만족하는 적어도 하나의 $m(m > n)$이 존재한다. 따라서 $S_{n+1}(i,j)$가 0이면 노드 (i,j)는 LIP의 원소이고, 1이면 LSP의 원소임을 알 수 있다. 본 논문은 계수에 대해 $S_{n+1}(i,j)$ 테스트 기법을 제안하며, MZP-DFS는 $S_{n+1}(i,j)$ 테스트의 결과에 따라 계수에 대한 인코딩 동작을 결정한다. $S_{n+1}(i,j)$ 테스트 기법은 SPIHT에서 LIP 및 LSP 리스트를 제거할 수 있을 뿐만 아니라 이전상태를 유지할 필요가 없기 때문에 추가의 저장

공간을 요구하지 않는 장점을 가진다.

노드 (i, j)를 루트로 하는 A type 트리를 $D(i, j)$, B type 트리를 $L(i, j)$라고 하고 노드 (i, j)의 4개의 자식(offsprings)을 $O(i, j)$을 라고 하자. SPIHT의 set partitioning 규칙에 따라 $D(i, j)$는 $S_n(D(i, j)) = 1$일 때 4개의 자식노드 $(k, l)(\in O(i, j))$과 (i, j)를 루트노드로 하는 하나의 $L(i, j)$로 분할되며, $L(i, j)$는 $S_n(L(i, j)) = 1$일 때 $(k, l) \in O(i, j)$를 루트노드로 하는 4개의 A type 서브트리 $D(k, l)$로 분할된다. 따라서, $D(i, j)$는 노드 (i, j)의 부모에 대한 $S_n(L(i/2, j/2)) = 1$ 일 때 LIS 리스트에 추가되며, $S_m(D(i, j)) = 0 (m > n)$일 경우 LIS에 존재한다. 즉, $S_{n+1}(D(i, j) = 0$이면 $D(i, j)$는 테스트 되어진다. $L(i, j)$에 대하여 동일한 논리를 적용하면 $S_n(D(i, j)) = 1$일 때 LIS 리스트에 추가되며, $S_{n+1}(L(l, j)) = 0$일 경우 테스트 되어짐을 알 수 있다. MZP-DFS는 트리에 대한 $S_{n+1}(T)$ 테스트 기법과 공간트리에 대하여 깊이 우선탐색을 수행함으로써 LIS 리스트를 제거한다.

$S_{n+1}(T)$ 테스트는 계수의 값을 기반으로 하기 때문에 복원 알고리즘에는 적용할 수 없다. SPIHT에서 리스트를 제거하기 위한 기존의 연구들은 픽셀에 대응되는 추가의 상태맵을 사용하고 있다. 점진적인 이미지 전송에 있어 이미지의 왜곡은 픽셀의 수 N에 대하여 $|c_{i,j}|^2/N$의 비율로 감소한다. 즉, 계수의 MSB에 대한 오류는 큰 이미지 왜곡을 유발하지만 LSB에 대한 오류는 왜곡의 정도가 매우 적음을 말해준다. 본 논문에서는 복원된 공간트리의 상태를 유지하기 위하여 추가의 상태맵을 사용하지 않고 계수의 LSB를 사용한 디코딩 알고리즘을 제안한다. MZP-DFS는 이미지 복원 과정에서 약 31%의 계수에 대해서 LSB 오류를 발생시킨다. 그러나 이미지의 왜곡에 미치는 영향은 매우 적으며, 실험에서 이미지의 시각적 품질이 SPIHT와 거의 동일함을 보이주고 있다. MZP-DFS는 LSB의 사용에 있어 효과적인 배치기법을 통하여 비트율이 높아질 경우 LSB의 오류가 점진적으로 복원되기 때문에 높은 비트율에서 완벽한 이미지 복원이 가능하다. MZP-DFS는 $D(i, j)$및 $L(i, j)$의 정보를 $(2i, 2j)$와 $(2i+1, 2j+1)$로 변환된 주소의 LSB에 저장함으로써 가장 작은 임계치에 대한 자식노드들의 처리과정에서 LSB의 사용으로 인한 오류를 복원한다.

MZP-DFS의 인코딩 알고리즘을 간략히 기술하면 (그림 3.1)과 같다.

```
for all head node (i, j)
    if (i, j) ∈ DCband b=0, else b=1
    EncodingPass (i, j, b)

EncodingPass (i, j, b)
    if S_{n+1}(i, j)=0
        output S_n(i, j)
        if S_n(i, j)=1
            output of Sign of (i, j)
    else
        output refinement bit of (i, j)
    if b=1
        if S_{n+1}(D(i, j))=0
            output S_{n+1}(D(i, j))
        if S_n(D(i, j))=1 and S_{n+1}(L(i, j))=0
            output S_n(L(i, j))
        if S_n(D(i, j))=1
            for all (k, l) ∈ O(i, j)
                EncodePass (k, l, S_n(L(i, j)))
```

(그림 3.1) MZP-DFS의 인코딩 알고리즘

MZP-DFS의 디코딩 알고리즘은 (그림 3.2)와 같으며, $B_0(i, j)$는 복원될 계수의 LSB에 대한 비트연산을 수행한다. 디코딩 과정에서 임계치가 $1 (n \geq 0)$일 때 부모노드에서 발생된 LSB 오류는 자식 노드에 대한 처리과정에서 수정된다.

```
for all head node (i, j)
    if (i, j) ∈ DCband b=0, else b=1
    EncodingPass (i, j, b)

EncodingPass (i, j, b)
    if n=0 B_0(i, j)=0
    if S_{n+1}(i, j)=0
        get bit
        if bit=1
            get sign bit and set (i, j) to ±2^n
    else
        get refinement_bit and modify (i, j)
    if b=1
        if B_0(2i, 2j)=0
            get bit and B_0(2i, 2j)=bit
        if B_0(2i, 2j)=1 and B_0(2i+1, 2j+1)=0
            get bit and B_0(2i+1, 2j+1)=bit
        if B_0(2i, 2j)=1
            for all (k, l) ∈ O(i, j)
                DecodePass(k, l, B_0(2i+1, 2j+1))
```

(그림 3.2) MZP-DFS의 디코딩 알고리즘

4. 실험 결과

MZP-DFS 알고리즘의 성능을 평가하기 위하여 메모리 사용량과 이미지 품질에 대한 실험을 하였다. 실험에 사용된 이미지는 512×512 크기의 Lena와 Barbara를 이용하였다. 실험에 사용된 웨이블릿 이미지는 Geoff Davis[7]의 라이브러리에서 Antonini[8] 9-7 탭 필터를 이용하였으며, 이미지 품질을 향상시키기 위한 추가적인 기법은 배제하였다.

〈표 4.1〉 비트율에 따른 메모리 요구량(byte)

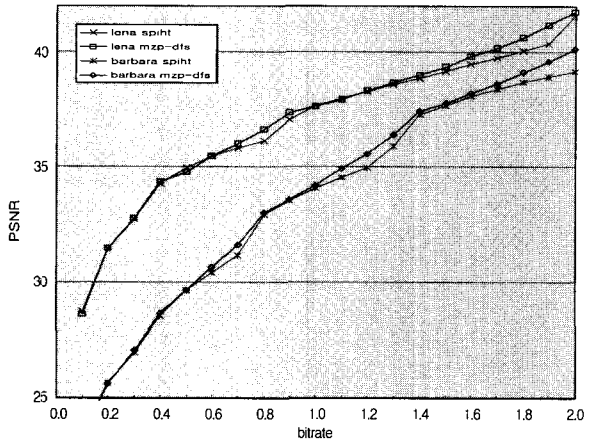
bitrate	MZP-DFS	SPIHT	NLS	SMAWZ
0.1	0 K	61 K	128 K	42 K
0.5	0 K	270 K	128 K	42 K
1.0	0 K	618 K	128 K	42 K
1.5	0 K	882 K	128 K	42 K
2.0	0 K	1222 K	128 K	42 K

<표 4.1>에서 MZP-DFS, SPIHT, NLS, SMAWZ에 대하여 추가로 요구되는 메모리 요구량을 비교하였다. SPIHT의 경우 리스트 한 개에 필요한 메모리 크기는 6바이트로 가정하였으며, LIS, LIP 및 LSP에 대한 각각의 최대 메모리 요구량을 합하였다. <표 4.2>는 이미지 크기에 따른 추가적인 메모리 요구량을 측정하였다.

〈표 4.2〉 이미지 크기에 따른 메모리 요구량

Image size	MZP-DFS	NLS	SMAWZ
256×256	0 K	32 K	10.5 K
512×512	0 K	128 K	42 K
1024×1024	0 K	512 K	168 K
2048×2048	0 K	2048 K	672 K

(그림 4.1)은 MZP-DFS와 SPIHT에 대하여 비트율에 의한 이미지의 품질을 측정한 결과이며, 공간트리의 루트 노드를 서칭하는 순서는 morton 방식을 사용하였다. (그림 4.1)에서와 같이 MZP-DFS 알고리즘은 이미지 품질면에서 SPIHT 알고리즘과 거의 동일함을 보여주고 있다. MZP-DFS는 복원될 계수에 대한 b_0 비트를 이용하기 때문에 부수적으로 이미지 품질 면에서 다소 향상된 결과를 보여주고 있다.



(그림 4.1) 비트율에 따른 PSNR

5. 결 론

SPIHT는 웨이블릿 변환에 기반한 이미지 압축기법으로 잘 알려져 있다. SPIHT는 제로트리의 특성을 이용한 set partitioning 규칙을 이용하여 이미지 압축을 수행하며, 이 과정에서 리스트 구조를 사용하고 있다. SPIHT에서 리스트 구조의 사용은 부가적인 메모리와 리스트를 관리하기 위한 논리를 요구하고 있어 하드웨어로 구현할 때 단점으로 지적되고 있다.

본 논문에서는 추가 메모리 요구 없이 SPIHT 알고리즘을 수행할 수 있는 MZP-DFS 알고리즘을 제안하였다. MZP-DFS는 SPIHT의 set partitioning 규칙을 사용하기 때문에 수행속도가 빠르고 효율적인 비트 스트림을 생성할 뿐만 아니라 SPIHT와 거의 동일한 이미지 품질을 보장한다. MZP-DFS 알고리즘의 수행에는 메모리가 소요되지 않으므로 하드웨어 구현에 있어 비용을 절감할 수 있으며, 각각의 공간트리를 독립된 프로세서로 처리할 수 있기 때문에 실시간 이미지 압축에 매우 적합하다.

참 고 문 헌

[1] Jerome M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," IEEE Trans. on Signal Processing, Vol.41, No.12, pp.3445-3462, December, 1993.

[2] Amir Said, William A. Pearlman, "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," IEEE Trans. CSVT, Vol.6, No.3.

[3] Frederick W. Wheeler, William A. Pearlman, "Low-Memory Packetized SPIHT Image Compression," Thirty-Third Annual Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, pp.24-27, October, 1999.

[4] Wen-Kuo Lin, Neil Burgess, "Listless Zerotree Coding for Color Images," in 32nd Asilomar Conference on Signals, System and Computers, CA, USA, November, 1998.

[5] R. Kutil, "A significance map based adaptive wavelet zerotree codec (SMAWZ)," In S. Panchanathan, V. Bove, and S. I. Sudharsanan, editors, Media Processors 2002, SPIE Proceedings, Vol.4674, pp.61-71, January, 2002.

[6] Frederick W. Wheeler, William A. Pearlman, "SPIHT Image compression Without Lists," IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2000), Istanbul, Turkey, June, 2000.

[7] http://www.geoffdavis.net/.

[8] M. Antonini, M. Barlaud, P. Mathieu and I. Daubechies, "Image coding using wavelet transform," IEEE Trans. Image Processing, Vol.1, pp.205-220, April, 1992.

[9] S. Mallat, "A theory for multiresolution signal decomposition : The Wavelet representation," IEEE Trans. Pattern Anal. Mach. Intell., Vol.11, pp.674-693, July, 1989.

김 충 길

e-mail : chgkim@chollian.net
1994년 부산대학교 전자계산학과(학사)
1996년 부산대학교 전자계산학과(이학석사)
2001년 부산대학교 멀티미디어협동과정 (박사수료)
1998년~2000년 창신대학 전임강사
2000년~2002년 성신외국어대학 전임강사
관심분야 : 영상 처리 및 압축, 멀티미디어 응용

이 주 경

e-mail : jklee@melon.cs.pusan.ac.kr
1996년 부산대학교 전자계산학과(학사)
1998년 부산대학교 대학원 전자계산학과 (이학석사)
1998년~2001년 한국전력공사 근무
2001년~현재 부산대학교 전자계산학과 박사과정
관심분야 : 멀티미디어, 동영상 코덱, 오류제어

정 기 동

e-mail : kdchung@melon.cs.pusan.ac.kr
1973년 서울대학교(학사)
1975년 서울대학교 대학원(이학석사)
1986년 서울대학교 대학원(이학박사)
1990년~1991년 MIT대학 교환 교수
1995년~1997년 부산대학교 전자계산소 소장
1999년~2001년 부산대학교 BK21 사업 단장
1978년~현재 부산대학교 전자계산학과 교수
관심분야 : 병렬처리, 멀티미디어