비디오객체의 경계향상을 위한 VLSI 구조

종신회원 김 진 상*

VLSI Architecture for Video Object Boundary Enhancement

Jinsang Kim* *Lifelong Member*

요 약

에지나 윤곽 정보는 인간의 시각 시스템에 의하여 가장 잘 인식되며 객체의 인식과 지각에 사용되는 중요한 정보이다. 그러므로 비디오내의 객체간의 상호작용, 객체기반 코딩과 표현과 같은 응용을 위하여, 비디오객체의 추출과정에 에지정보를 적용하면 인간의 시각 시스템과 근접한 객체 경계를 얻을 수 있다. 대부분의 객체추출 방식은 연산량이 많고 반복적인 연산을 수행하므로 실시간 처리가 어렵다. 본 논문에서는 비디오객체 분할 과정에 에지 정보를 적용하여 정확한 객체 경계를 추출하는 VLSI 구조를 제안한다. 제안된 하드웨어 구조는 연산방식이 간단하므로 하드웨어로 쉽게 구현될 수 있으며, 제안된 VLSI 하드웨어 구조를 이용하면 객체기반 멀티미디어 응용을 위하여 실시간으로 비디오객체를 분할할 수 있다.

Key Words : edge fusion, video object, segmentation, systolic architecture, VLSI.

ABSTRACT

The edge and contour information are very much appreciated by the human visual systems and are responsible for our perceptions and recognitions. Therefore, if edge information is integrated during extracting video objects, we can generate boundaries of objects closer to human visual systems for multimedia applications such as interaction between video objects, object-based coding, and representation. Most of object extraction methods are difficult to implement real-time systems due to their iterative and complex arithmetic operations. In this paper, we propose a VLSI architecture integrating edge information to extract video objects for precisely located object boundaries. The proposed architecture can be easily implemented into hardware due to simple arithmetic operations. Also, it can be applied to real-time object extraction for object-oriented multimedia applications.

Ⅰ. 서 론

비디오내의 객체를 추출하기 위한 알고리즘 [1,2]이 제안되었으나, 이들은 일반적으로 비디오 장면의 텍스추어 (texture) 영역을 고려하여 처리하지 않으며 대부분의 비디오객체 추출을 위한 영역추출 알고리즘은 직렬형의 반복적인 연산 방식을 적용하여 많은 계산량이 필요하다. 이들 알고리즘의 대표적인 예는 K-means, RSST(Recursive Shortest Spanning Tree)[1]과 fuzzy C-means[2]이다. 본 논문에서는 기존의 알고리즘보다 연산방식이 간단하고 하드웨어 구현이 용이한 저자가 제안한 알고리즘[8]을 하드웨어로 구현할 수 있는 VLSI 구조를 제안한다.

제안된 알고리즘의 흐름도는 그림 1과 같다. 적용된 알고리즘은 먼저 비디오 신호에서 각 화소 마다 움직임 벡터, 텍스추어, 색상정보 등의 다중 특징 벡터(multiple feature vector) 를 추출한다. 추출된 특징 벡터는 필터링, 정규화 및 가중화 과정을 거쳐 SOFM(Self-Organizing Feature Maps) 신경망[3]으로 입력된다. 신경망은 입력된 특징벡터와 학습

* 경희대학교 전자공학과 (jskim27@khu.ac.kr)
논문번호 : KICS020288-0703, 접수일자 : 2002년 7월 3일

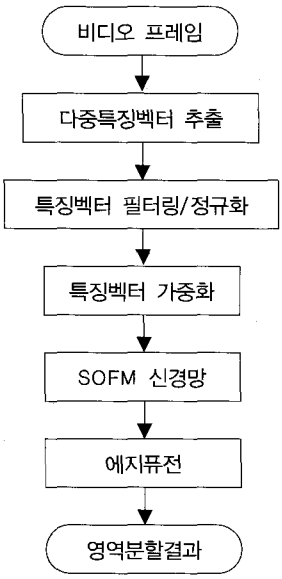

그림 1. 영역분할 알고리즘의 흐름도

과정을 거쳐 이미 완성된 코드북 내의 가장 근접한 코드벡터를 계산하여 이의 인덱스를 영역분할의 레이블로 이용한다. 신경망의 출력은 대부분 과도영역분할(over-segmentation) 결과이므로, 영역 합병(region merging) 과정을 거쳐 최종적인 영역분할 결과를 얻게 된다. 영역합병 과정에 에지(edge) 정보를 조합하면 인간의 시각정보에 근접한 의미 있는 영역분할 결과를 얻을 수 있다.

적용된 알고리즘은 기존의 방법보다 영역의 개수와 영역내의 특징 분산값의 곱을 평가 기준치[4]로 적용할 경우, 최대 60% 이상의 성능향상을 얻을 수 있다. 본 논문에서는 적용된 알고리즘에 대하여 각 기능 블록별로 수행시간을 분석하여 알고리즘의 병목 부분인 에지퓨전(edge fusion) 과정을 하드웨어로 매핑시켜 영역분할 알고리즘을 실시간으로 처리할 수 있도록 하는 에지퓨전 하드웨어 구조를 제안한다.

제안된 영역분할 알고리즘의 각 모듈별 실행시간을 관찰하고 비교하기 위하여 하나의 프레임에 대한 영역분할 알고리즘을 수행할 때 필요한 평균 CPU 시간 데이터를 수집하였다. 컬러 QCIF(176x144) 비디오 입력에 대한 영역분할 알고리즘을 500 MHz PC상에서 동작되는 MATLAB 코드를 이용하여 구현할 경우, 509 초의 시간이 필요하고 각 모듈 별 CPU 수행시간은 표 1과 같이 영역분할의 후처리 과정이라 할 수 있는 에지퓨전 모듈은 전체의 CPU 시간의 77%를 차지한다.

표 1. 영역분할 알고리즘의 수행시간 분석

subtask	CPU time(초)	%
Feature Extraction	94.2	18.6
Feature Filtering	17.4	3.5
Feature Normalization	0.1	0.02
Feature Weighting	0.1	0.02
Feature Labeling	6.25	1.3
Edge Fusion	390.4	76.8
Total	509	100

특징벡터 추출과정은 19% 정도의 CPU 시간 점유율을 차지하는데 이는 움직임 벡터를 추출하는데 많은 연산량이 필요하기 때문이다. SOFM 신경망 회로를 이용한 특징벡터 레이블링 과정은 전체의 1.3% 정도의 매우 작은 CPU 시간 점유율을 차지하는데, 이는 신경망을 이용한 처리과정은 병렬처리가 가능하고 승산과 가산 등의 간단한 연산만이 필요하기 때문이다. 표 1은 실시간 영역분할을 위해서는 가장 많은 CPU 수행시간을 요구하는 에지퓨전 모듈을 특정한 하드웨어로 구현해야 함을 명확하게 보여준다.

Ⅱ. 에지퓨전 하드웨어 구조

에지퓨전 과정은 그림 2와 같이 요약할 수 있다. 먼저, SOFM 신경망 출력인 초기 영역분할 결과와 분할하고자 하는 에지 정보를 결합하여 결합요소분석(CCA: connected component analysis)[5]를 이용하여 에지정보가 결합된 영역분할 레이블을 구한다. 에지를 결합할 때, 분리된 에지를 연결하여(edge linking)[6]보다 많은 영역이 연결될 수 있도록 한다.

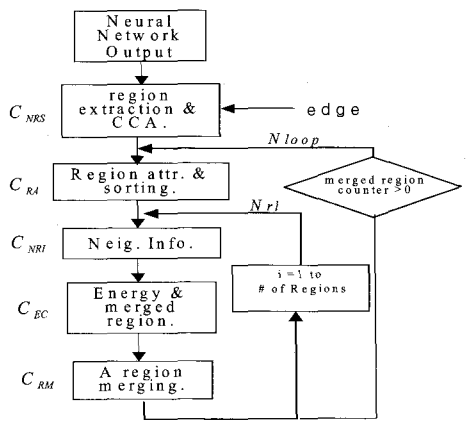

그림 2. 에지퓨젼 과정

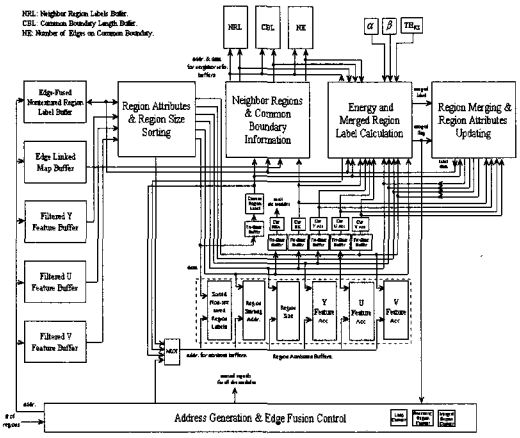

그림 3. 에지퓨전 하드웨어의 블록도

영역합병 과정은 레이블링된 영역의 크기가 작은 것부터 연속적으로 수행되는데, 이를 위하여 각각의 영역의 속성을 구한다. 현재 처리되고 있는 영역의 이웃영역과의 기하학적인 정보와 에지정보를 계산하여 이를 영역합병 판단 기준인 기준영역과 이웃영역과의 에너지를 계산하는데 이용할 수 있도록 한다. 에너지 값이 임계치 보다 작고 최소의 에너지 값을 갖는 이웃영역이 기준영역과 합병된다. 더 이상 합병되는 영역이 존재하지 않으면 에지퓨전 과정을 종료한다(이때 Nrl은 영역갯수, Nloop는 반복되는 loop 개수임).

그림 3은 컬러 비디오에 대하여, 상기와 같은 에지퓨전 과정을 수행하는 하드웨어 구조를 간략하게 나타낸 것이다. 에지가 결합된 영역 레이블 정보, 에지링킹 맵(edge linking map), 미디언 필터링(median filtering) 된 3개의 YUV 특징은 5개의 로컬 버퍼에 저장된다. 영역속성과 정렬 모듈은 각 영역의 크기, 시작번지, YUV의 누적값을 계산한다. 또한 작은 크기의 영역부터 영역합병 과정을 수행하기 위하여 영역의 크기를 정렬하여 이를 버퍼에 저장한다. 현재 처리되고 있는 영역의 이웃 영역 레이블 값, 공통된 경계 길이, 공통된 경계상의 에지 갯수는 각각 NRL, CBL, NE 레지스터에 저장된다.

현재 처리되고 있는 영역과 이웃영역과의 합병은, 현재 영역과 이웃영역과의 속성정보를 이용한 에너지 함수를 구하여 가장 작은 에너지 값을 같은 이웃영역과 합병이 이루어진다. 이러한 과정은 에지퓨전 제어모듈에 의하여 이루어지며, 이 모듈은 언제 영역합병과정이 종료되어야 하는지를 결정한다. 다음은 에지퓨전 구조의 각 모듈별 상세한 기능을 설명한다.

2.1 에지 정보를 이용한 영역 분열

텍스추어 특징을 띠고 있지 않은 영역에 대하여 에지퓨전 알고리즘을 수행하며 그림 4는 이러한 과정을 수행하는 하드웨어 모듈이다.

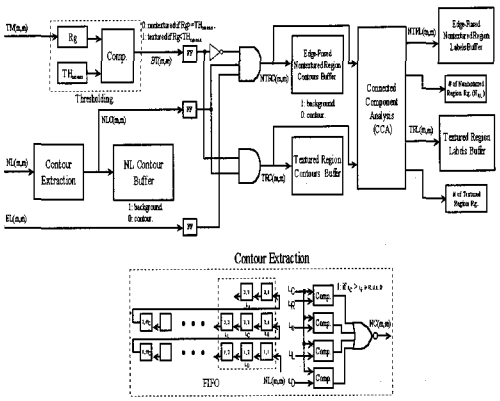

그림 4. 텍스추어 영역 추출과 에지를 이용한 영역 분열

신경망 모듈의 출력 레이블 값을 FIFO의 입력으로 받아들여 영역의 경계(NC)를 추출한다. 이후 3개의 버퍼에 있는 특징(texturedness, 윤곽(contour), linked edge map)을 픽셀 단위로 스캔하여 이를 논리 게이트에 입력하여 에지 정보가 결합된 윤곽을 생성한다. 이때 texturedness 측정값(TM)이 임계값, THtmeas 보다 작으면 2진texturedness 값(BT)은 1이다. 에지 정보가 결합된 영역 레이블(NTRL)은 영역의 2진 윤곽을 결합요소분석 모듈을 통하여 얻어진다. (m,n)은 프레임내의 하나의 화소 위치를 의미한다.

에지퓨전 제어기는 신경망 출력값(레이블)을 저장한 버퍼를 좌에서 우로, 위에서 아래로 스캔하여 FIFO에 입력시킨다. 만일 현재 화소 위치를 기준으로 하여 상하좌우 4개의 이웃화소 레이블 값(LR, LU, LL,LD) 이 현재의 화소 위치의 레이블값(LC) 보다 크면 현재의 화소 위치의 2진 윤곽 값은 0 이고 그 외는 1 이다. 에지 정보가 결합된 2진 윤곽의 레이블 값을 구하기 위하여 [6]에서 제시한 하드웨어 구조를 적용한다. 이 모듈은 클럭 수(CNRS)는 윤곽추출(*NC+1+N*), 에지 정보를 결합한 영역분열(N)과 결합요소 하드웨어(3N 클럭이 필요)에 필요한 클럭 수의 합이다.

2.2 영역 속성과 영역 크기 정렬 모듈

영역 합병은 작은 크기의 영역부터 시작해야 화소의 색상 값의 차이로 인한 영향을 덜 받는다. 그

러므로 각 영역의 크기를 정렬하는 모듈이 필요하다. 영역합병을 하기 위한 기준 영역과 그 이웃 영역간의 에너지 값을 이용하여 기준영역이 이웃영역들 중 하나와 합병될 수 있는지를 판별한다. 이와 같은 이유로 인하여 각 영역의 크기, YUV 누적값 등의 속성을 계산한다. 또한 한 프레임내의 기준 영역과 이웃영역이 차지하는 화소의 위치를 빠르게 스캔하기 위하여 각 영역의 시작 어드레스를 계산하여 보관한다. 이는 레이블 값의 preloaded 어드레스로 사용된다.

그림 5는 영역의 속성과 영역 크기를 정렬하는 모듈의 구조이다. 스캔된 레이블과 3개의 컬러 특징 누적값이 계산되어 해당 버퍼에 저장되는 동시에 영역의 시작 어드레스가 계산된다. 영역의 속성값 계산은 가산기를 이용하여 쉽게 구현할 수 있다. 영역 크기의 정렬을 위하여 log(NRL)-프로세서 heap 정렬 구조[7]를 적용하는데 이는 O(NRLlog(NRL)) 클럭 사이클이 필요하다. 이 모듈을 수행하는데 필요한 클럭 수(CRA)는 N+NRLlog(NRL)이다.

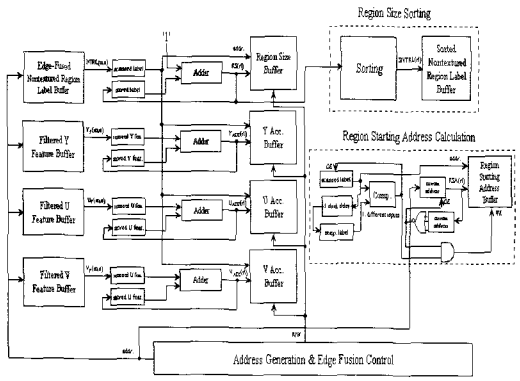

그림 5. 영역속성 계산 모듈

2.3 영역간의 경계정보 계산 모듈

이 모듈은 기준영역과 이웃영역 간의 경계정보는 에너지 값을 계산하는데 필요하다. 영역간의 경계정보는 기준영역에 대한 이웃영역의 레이블 값, 현재 영역과 이웃 영역간의 공통 경계 길이, 공통 경계상의 에지의 숫자이다.

그림 6은 영역간의 경계정보 계산 모듈이다. 스캐닝된 영역 레이블 값은 FIFO에 입력되고, 기준 영역 레이블 값과 4개의 이웃의 레이블 값(NRL: Neighbor Region Labels)이 계산된다. 동시에 공통 경계 길이(CBL: Common Boundary Length) 와 에지 개수(NE: Number of edges)도 계산된다. 이 모듈을 구성하기 위하여 필요한 메모리 크기는, NRLi

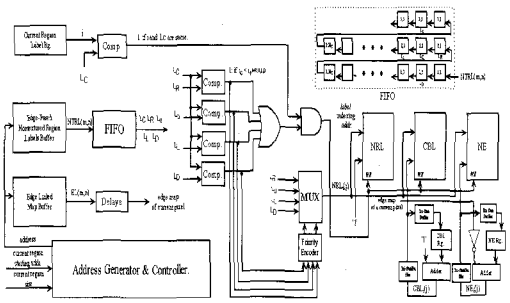

그림 6. 영역간의 경계정보 계산 모듈.

(j) 계산을 위한 NRL x 1 비트, CBLi(j)와 NE 계산을 위한 2개의 NRL x log(NRP) 비트이다. 여기서 NRP 는 분할된 프레임내의 영역 경계선의 최대 길이를 의미한다. 또한 이 모듈에 필요한 클럭 수(CNRL)는 (RegL+2)NC+(NC+2)이다. NC와 RegL은 각각 한 프레임의 열 개수와 기준 영역의 길이(한 프레임 내에서 차지하는 행의 개수) 이다.

2.4 이웃 영역과의 에너지 계산 모듈과 영역합병 및 영역속성 갱신모듈

에너지 계산 모듈은 기준영역과 이웃영역 간의 에너지 계산기능 및 이웃영역 중에서 기준영역과 합병될 수 있는 영역이 존재하는지의 여부를 판단하는 모듈이다. 이 모듈의 하드웨어 구조를 설계하여 분석한 결과, 에너지 계산을 위한 하드웨어 모듈은 비교기, 가산기, MUX와 제산기를 이용하여 구현될 수 있고 이 모듈에 필요한 클럭 수(CEC)는 최대 NRL 됨을 확인하였다.

영역합병 및 영역속성 갱신 모듈은 기준영역과 이웃영역을 합병하고 합병된 새로운 영역에 대한 속성을 갱신하여 다음의 반복 연산과정에 이용되도록 한다. 기준영역을 스캐닝하는 시간을 줄이기 위하여 기준영역의 시작 어드레스 값과 영역 크기값이 이용된다. 합병후 영역속성을 갱신하기 위하여, YUV 컬러 특징 평균값 대신 누적값을 저장하여 합병될 두 영역의 누적값의 가산 결과를 합병된 영역의 크기 값으로 나누어서, 합병된 영역의 컬러 평균값을 쉽게 계산할 수 있도록 하였다. 이 모듈은 가산기와 비교기를 이용하여 쉽게 구현할 수 있다. 합병되기 전 두 영역의 시작 어드레스 중 작은 값이 합병된 영역의 시작 어드레스 값으로 갱신되는데 이는 스캐닝 순서가 프레임의 상위 열부터 시작되기 때문이다. 이 모듈의 클럭 수(CRM)의 최대값은(merged flag 비트 값)*RegL*Nc이고 merged flag 값이 0 이면 이 모듈은 작동되지 않는다.

Ⅲ. 성능분석

에지퓨전을 수행하기 위한 최대 클럭수는 CNRS+NLOOP(CRA+NRL(CNRI+CEC+CRM))이다. 여기서 Nloop는 에지퓨전 과정에 필요한 최대 loop 수를 의미한다. 비디오 입력(형식: QCIF, 256 x 256)에 대한 시뮬레이션을 통하여 산출된 최악의(worst case) 파라미터(NRL, RegL, NLOOP)를 이용하여 제안된 에지퓨전 하드웨어의 동작속도와 필요한 하드웨어 요소(hardware components) 를 표 2와 표 3과 같이 예측하였다.

표 2. 에지퓨전 하드웨어 동작 속도

QCIF	431	3.57	3.83	1.15e7
256x256	649	4.41	4.41	2.09e7

표 3. 에지퓨전 하드웨어 요소

QCIF	2.34 M	489 (6.71)	18 (9.34)	7 (23.00)	29 (10.89)
256x 256	5.51 M	769 (7.02)	18 (9.62)	7 (24.00)	30 (12.34)

영역의 개수(NRL), 영역의 길이(RegL)와 반복횟수(Nloop)는 에지퓨전 시뮬레이션 과정에서 얻어지는 값을 평균하였다. 표 2와 같이 QCIF 한 프레임에 대한 에지퓨전을 실행할 경우 최대 클럭수는 1.15e7이므로 287MHz의 클럭을 사용하면 30프레임/초의 QCIF 이미지에 대하여 실시간으로 에지퓨전을 수행할 수 있다. 이 경우 제안된 하드웨어는 소프트웨어보다 월등한 성능향상을 얻을 수 있음을 의미한다.

에지퓨전 알고리즘의 하드웨어 구현에 필요한 하드웨어 요소를 분석하기 위하여 메모리, 레지스터, 가산기(감산기), 승산기(제산기), 비교기의 개수와 평균 데이터 폭(표 3의 괄호안의 값) 을 표 3과같이 추정하였다. 가산기, 승산기, 비교기와 같은 조합논리 소자의 하드웨어 복잡도는 입력 비디오 프레임의 크기에 영향의 적게 받으며 메모리 소자는 비디오 프레임의 크기에 영향을 많이 받는다. 이는 프레임의 크기가 증가하면 임시 결과를 저장하는 버퍼와 레지스터의 크기가 증가해야 하기 때문이다.

Ⅳ. 결 론

본 논문에서는 적용된 알고리즘의 수행과정에 필요한 모듈 중에서 가장 많은 시간을 소모하는 병목 모듈인 에지퓨전 모듈에 대한 ASIC 구현 방법을 제안하였다. 제안된 하드웨어 구조는 버퍼의 스캐닝 시간을 줄이기 위하여 각 영역의 속성을 계산할 때 영역의 시작 번지도 계산할 수 있도록 하였다. 영역의 레이블을 이용한 인덱스 어드레싱의 경우 처음부터 레이블 버퍼를 스캐닝하지 않고 바로 시작번지로 이동하여 레이블 값을 스캐닝 할 수 있게 하였다. 또한 영역을 합병한 후, 각 영역의 속성을 갱신해야 하는데 특징(YUV 컬러 값)의 평균값 대신 누적값과 영역 크기를 저장한다 실제적인 컬러 특징의 영역 평균값은 에너지 계산 모듈에서 누적값과 영역의 크기를 제산하여 계산되어 에너지를 구할 때 이용된다. 이와 같이 컬러 특징 누적값을 속성으로 이용하면 두 영역간의 특징 차이를 계산할 때 빠르게 연산을 수행할 수 있다.

제안된 VLSI 구조의 성능과 하드웨어 복잡도를 영역분할 수행과정에서 추출된 최악의 파라미터를 이용하여 예측하였다. 30 프레임/초로 입력되는 QCIF 크기의 비디오는 287 MHz 클럭을 이용하면 실시간으로 에지퓨전을 수행할 수 있다. 대부분의 비디오는 프레임간 상관관계(temporal correlation)가 높아 MPEG-4와 같은 멀티미디어 데이터의 객체기반 코딩 응용을 위하여 비디오의 객체를 추출하고자 할 때, 모든 프레임에 대한 영역분할 알고리즘을 수행할 필요가 없다. 어떤 응용을 위하여 f 프레임 간격을 두고 영역분할을 수행하여도 되는 경우, 제안된 알고리즘은 프레임 크기가 QCIF 보다 큰 비디오에 대하여도 실시간으로 에지퓨전을 수행할 수 있다. 예를 들어 f=4인 경우 CIF(QCIF의 4배의 프레임 크기) 비디오 입력에 대하여도 실시간으로 처리가 가능하다. 예측된 하드웨어의 복잡도가 크지 않으므로, 에지퓨전 하드웨어를 ASIC 형태로 쉽게 제작할 수 있다.

참 고 문 헌

[1] A.A.Alatan and et al. "Image Sequence Analysis for Emerging Interactive Multimdia Services-The European COST211 framework" IEEE Trans. CSVT, vol 8. 802-813, Nov. 1998.

[2] Roberto Castagno and et al., "Video Segmentation Based on Multiple Features for Interactive Multimedia Applications," IEEE

Trans. CSVT, vol 8, no. 5, pp. 562-571, Sep. 1998.

[3] T. Kohonen, "Self-organized formation of topologically correct feature maps," Biological Cybernetics, vol. 4, pp.59-69, Jul. 1982.

[4] Y. J. Zhang, "A Survey on Evaluation Methods for Image Segmentation," Pattern Recognition, vol. 29, no.8, pp. 1335-1346, 1996.

[5] Xue Dong Yang, "Design of Fast Connected Components Hardware," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 937-944, 1996.

[6] Amjad and T. Chen, "A VLSI Architecture for Real-Time Edge Linking," IEEE Trans. PAMI, Vol. 21, no.1, pp. 89-94, 1999.

[7] C. D. Thompson, "The VLSI Complexity of Sorting," IEEE Trans. on Computers, vol 32, pp. 1171-1184, 1983.

[8] Jinsang Kim and T. Chen, "Low-Complexity Fusion of Intensity, Motion, Texture and Edge for Image Sequence Segmentation: A Neural Network Approach," IEEE International Workshop on Neural Networks for Signal Processing, Sydney, Australia, pp.497-506, Dec. 2000.

김 진 상 (Jinsang Kim) 종신회원

2000년 12월 미국 콜로라도 주립대 공학박사
1990년 2월~2001년 8월 KT 연구소
2001년~현재 경희대학교 전자정보학부 조교수
<관심분야> 영상처리와 이동통신용 SoC 설계