디지털융복합연구
제12권 제6호

ISSN : 1738-1916(Print)

처리 속도 향상을 위해 OpenCV CUDA를 활용한 도로 영역 검출

이태희, 황보현, 윤종호, 최명렬

To cite this article : 이태희, 황보현, 윤종호, 최명렬 (2014) 처리 속도 향상을 위해 OpenCV CUDA를 활용한 도로 영역 검출, 디지털융복합연구, 12:6, 231-236

http://dx.doi.org/10.14400/JDC.2014.12.6.231

처리 속도 향상을 위해 OpenCV CUDA를 활용한 도로 영역 검출

이태희[*], 황보현[**], 윤종호[***], 최명렬[****]
한양대학교 전자통신공학과[*], 한양대학교 전자전기제어계측공학과[**],
한양대학교 전자통신전파공학과[***], 한양대학교 전자통신공학과[****]

A Road Region Extraction Using OpenCV CUDA To Advance The Processing Speed

Tae-Hee Lee[*], Bo-Hyun Hwang[**], Jong-Ho Yun[***], Myung-Ryul Choi+
Dept. of Electronics & Communication Engineering, Hanyang University[*]
Dept. of EECI Engineering, Hanyang University[**]
Dept. of Electrical & Computer Engineering, Hanyang University[***]
Dept. of Electronics & Communication Engineering, Hanyang University+

요 약 본 논문은 호스트(PC) 기반의 직렬처리 방식으로 도로영역 추출 방식에 디바이스(Graphic Card) 기반의 병렬 처리 방식을 추가함으로써 보다 향상된 처리 속도를 가지는 도로영역검출을 제안하였다. OpenCV CUDA는 기존의 OpenCV와 CUDA를 연동하여 병렬 처리 방식의 많은 함수들을 지원한다. 또한 OpenCV와 CUDA 연동 시 환경설정이 완료된 OpenCV CUDA 함수들은 사용자의 디바이스(Graphic Card) 사양에 최적화된다. 따라서 OpenCV CUDA 사용은 알고리즘 검증 및 시뮬레이션 결과 도출의 용이성을 제공한다. 제안된 방법은 OpenCV CUDA 와 NVIDIA GeForce GTX 560 Ti 모델의 그래픽 카드를 사용하여 기존 방식보다 3.09배 빠른 처리 속도를 가짐을 실험을 통해 검증한다.

주제어 : GPU, CUDA, OpenCV CUDA, 병렬 처리

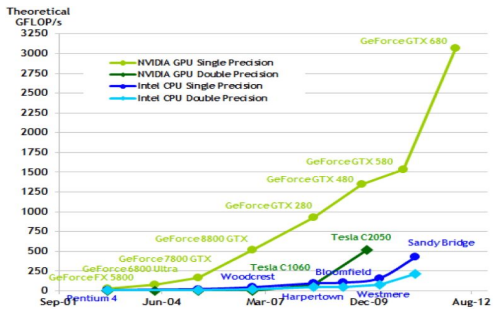
Abstract In this paper, we propose a processing speed improvement by adding a parallel processing based on device(graphic card) into a road region extraction by host(PC) based serial processing. The OpenCV CUDA supports the many functions of parallel processing method by interworking a conventional OpenCV with CUDA. Also, when interworking the OpenCV and CUDA, OpenCV functions completed a configuration are optimized the User's device(Graphic Card) specifications. Thus, OpenCV CUDA usage provides an algorithm verification and easiness of simulation result deduction. The proposed method is verified that the proposed method has a about 3.09 times faster processing speed than a conventional method by using OpenCV CUDA and graphic card of NVIDIA GeForce GTX 560 Ti model through experimentation.

Key Words : GPU, CUDA, OpenCV CUDA, Parallel processing

Received 4 March 2014, Revised 14 April 2014
Accepted 20 June 2014
Corresponding Author: Myung-Ryul Choi(Dept. of Electronics & Communication Engineering)
Email: Choimy@hanyang.ac.kr
ISSN: 1738-1916

1. 서론

그래픽 카드(Graphic Card)는 그래픽 프로세서(GPU, Graphics Processing Unit)와 메모리를 가지고 있으며, 컴퓨터에서 요구되는 그래픽 관련 부동 소수점 연산들을 주로 담당한다. [Fig. 1]은 그래픽 프로세서(GPU)와 인텔 CPU의 연산 처리 능력을 비교한 것으로 이미 GPU의 성능은 CPU를 압도하고 있다.



[Fig. 1] Comparison of computational power for NVIDIA graphic card and CPU

GPGPU(General Purpose computations on GPU)가 수학 및 과학 분야에서 급속히 확산되고 있다. 특히, GPU의 구조는 작은 프로세서의 배열로 되어 있기 때문에 병렬 연산에서 탁월한 성능을 보여주고 있다. 행렬 연산, 데이터 정렬 등 기본 연산에서 이미 CPU보다 고속 처리가 가능하며, 다체 문체, 바이오 정보처리 등의 응용에서도 효과적임을 입증하는 논문이 다소 발표되었다[1].

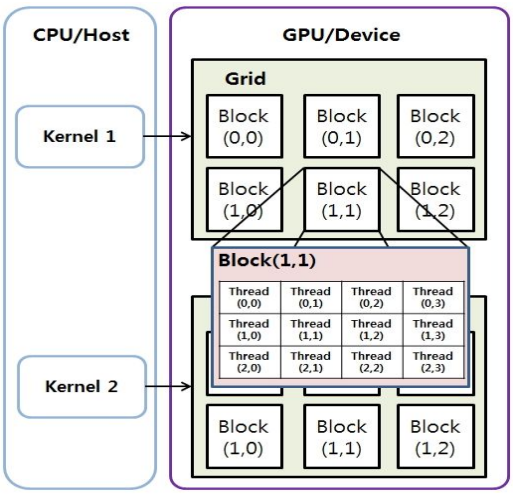
본 논문의 구성은 다음과 같다. 2장에서 CUDA 병렬처리 구조와 OpenCV CUDA에 대하여 기술하고, 3장에서는 기존 CPU상에서의 도로영역검출 알고리즘과 제안하는 CUDA를 활용한 도로영역검출 알고리즘을 비교하였다. 4장에서는 실험 결과를 보여주고, 마지막으로 5장에서 결론을 맺는다.

2. 병렬처리

CUDA는 NVIDIA의 병렬 컴퓨팅 아키텍처로 뛰어난 GPU 성능을 이용해 컴퓨팅 성능을 극적으로 높여주며. 현재까지 수백만 개의 CUDA 지원 GPU가 판매되고 있다. 또한 CUDA는 OpenCV와 연동되어 개발자로 하여금 보다 개선된 개발 환경을 제공한다.

2.1 CUDA 병렬처리 시스템

GP-GPU라 불리는 기술은 빠른 속도로 대량의 데이터를 처리해야 하는 비디오 인코딩이나 컴퓨터비전 분야 등에서 실시간적인 성능 보장을 위해 사용되고 있다. CUDA는 C언어를 이용하여 GPU에서 범용 컴퓨팅을 위해 설계된 하드웨어와 소프트웨어 구조를 말한다. CUDA를 이용한 고속 도로영역 검출은 [Fig. 2]와 같은 모델 기반의 OpenCV 연동을 통해 수행된다. CUDA에서 실행될 커널(kernel) 함수가 준비되면, 커널을 호출하기 전 그리드(grid)를 구성해야 한다. 그리드는 다시 각각의 블록(block)으로 구성되고, 하나의 블록은 많은 수의 스레드를 생성하고 서로 데이터를 공유하며 병렬처리를 수행하게 된다[2].



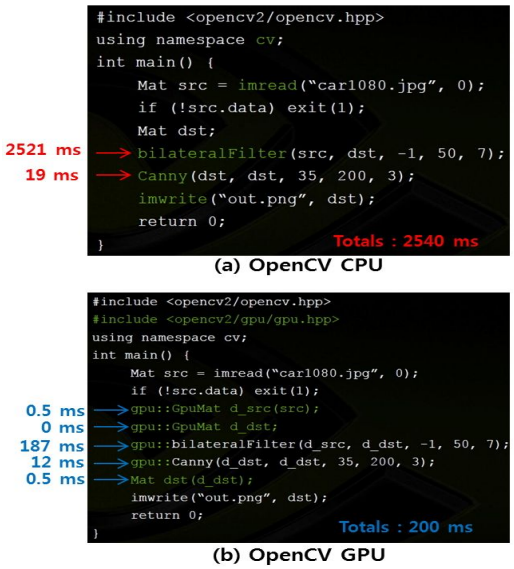
[Fig. 2] Heterogeneous programming model with CUDA

2.2 OpenCV CUDA

현재 OpenCV 프로그램은 CUDA와 연동하여 OpneCV 상에 GPU Module을 지원한다. OpneCV GPU Module의 목표는 GPU를 가지고 최상의 성능을 달성하는 것이다. 따라서 OpenCV GPU Module은 GPU 구조에 대하여 효율적인 커널(kernel) 생성과 비동기 실행(asynchronous execution), 복사 중복(copy overlaps), 제

로 복사(zero-copy) 등의 최적화된 데이터 흐름(Optimized Data Flows)을 제공한다. 또한 다양한 OpenCV GPU 함수들을 제공하고 있다.

[Fig. 3]은 OpenCV 상에서 양방향 필터 처리와 캐니 검출을 OpenCV Host(CPU) 환경과 OpenCV Device(GPU, Graphic Card) 환경에서의 함수 사용 및 처리 속도를 비교한 것이다. OpenCV GPU(CUDA) 처리 속도가 OpenCV CPU 처리 속도보다 약 12.7배 정도의 빠른 처리 속도를 가짐을 확인 할 수 있다.



[Fig. 3] Comparison of speed for CPU and GPU

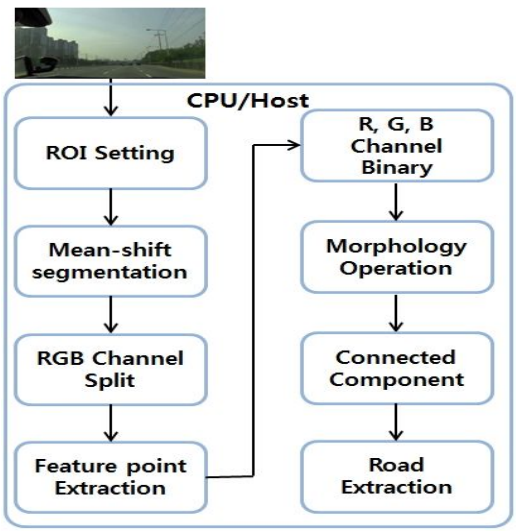
3. 도로영역검출

무인 드라이빙 시스템이나 차량의 비전 시스템에 있어서 도로 영역 검출은 중요하다. 현재 비전에 기반으로 하는 도로 영역 검출에 대한 많은 연구가 수행되고 있다[3]. 지능형 자동차 분야에 대한 영상처리는 정확성(또는 신뢰성)과 신속성을 바탕으로 한다면 본 논문에서 제안하는 방식은 영상처리의 신속성에 비중을 두고 있다.

3.1 기존의 도로영역검출

기존의 도로영역검출 방식은 입력영상에 대하여 관심영역을 설정하고, 관심영역에 Mean-Shift Algorithm을 수행하여 컬러분할을 수행한다. 분할된 영상에서 도로영역의 특징점(Feature Point)을 추출하고, 추출한 특징점을 이용하여 도로영역을 검출한다.

특징점을 이용하여 도로영역을 검출할 경우, 잡음 및 도로영역 이외의 영역도 검출되므로 모폴리지 연산과 연결된 구성요소를 수행하여 잡음을 제거하고 가장 큰 외곽선을 찾아 도로영역으로 설정한다. [Fig. 4]는 CPU기반의 도로영역검출을 순차적으로 수행되는 과정을 나타낸다[4].



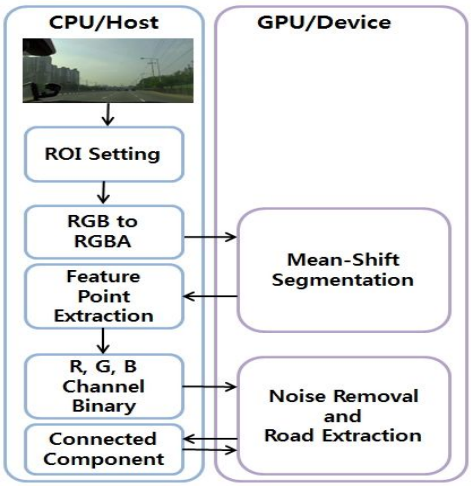
[Fig. 4] CPU Based Road Region Extraction

3.2 제안하는 도로영역검출

본 논문에서 제안하는 도로영역검출 알고리즘은 기존의 알고리즘 기반에 OpenCV CUDA를 활용하여 보다 빠른 처리 속도를 제안한다. [Fig. 5]는 OpenCV CUDA를 활용하여 병렬 처리 방식인 고속의 도로영역검출 알고리즘을 나타낸다.

제안하는 알고리즘은 기존의 CPU기반의 도로영역검출 알고리즘의 8개의 순서도 중에서 Mean-Shift Segmentation, RGB channel split, Morphology operation, Road extraction의 4개의 순서에 대해 CUDA 기반의 병렬 처리를 수행한다.

4개의 순서도에 대한 병렬처리는 OpenCV CUDA 함수를 사용하여 구현하였다. <Table 1>은 기존의 도로영역검출에 사용되었던 CPU기반의 함수와 병렬 처리가 가



[Fig. 5] Road Region Extraction using GPU

능한 OpenCV CUDA 함수를 비교한 표이다.

일반적으로 OpenCV Host(CPU) 상에서는 Mat를 생성하여 데이터 처리를 하지만 OpenCV CUDA 상에서는 OpenCV Host 상에서 생성한 Mat 구조를 사용할 수 없으므로 gpu::Mat을 생성하여 Device(GPU, Graphic Card) 내부에서 병렬로 영상을 처리하게 된다. OpenCV CUDA에서는 Upload 명령어를 사용하여 OpenCV Host에서 생성한 Mat 데이터를 Device의 gpu::Mat 데이터로 전송하고, Download 명령어를 사용하여 Device 내부에서 처리된 gpu::Mat 데이터를 OpenCV Host의 Mat 데이터로 전송함으로써 Host와 Device 사이에 데이터 전송이 이루어진다.

〈Table 1〉 Comparison of function

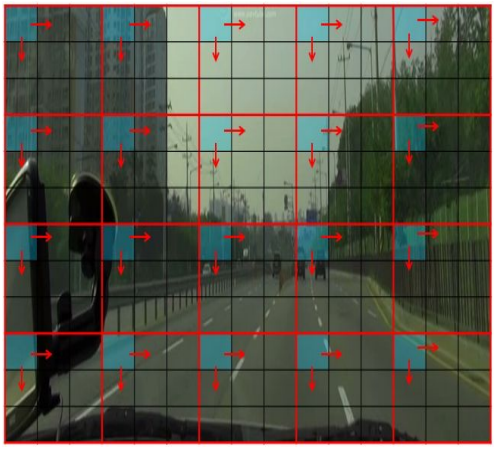
OpenCV CPU Function	OpenCV CUDA Function
cv::Mat	cv::gpu::Mat
cv::pryMeanShiftFiltering	cv::gpu::meanShiftFiltering
cv::split	cv::gpu::split
cv::morphologyEx	cv::gup::morphologyEx
cv::bitwise_and	cv::gpu::bitwise_and

OpenCV CUDA는 GeForce GTX 560 Ti 모델의 그래픽 카드에서 병렬처리를 수행하게 된다. <Table 2>는 GeForce GTX 560 Ti에 대한 구체적인 그래픽 카드 사양을 나타내며, GeForce GTX 560 Ti는 Grid당 최대 65535개의 Block을 생성할 수 있으며, Block당 최대 1024개의 Thread를 생성할 수 있다. 따라서 GeForce GTX 560 Ti 모델의 그래픽카드는 GPU Core의 연산을 위한 최소한의 단위로 구성되는 Thread를 최대 67,107,840 (65,535x1,024) 개를 생성할 수 있다. 또한 GeFoce GTX 560 Ti 모델은 8개의 SM(Streaming Multiprocessor)를 가지고 있으므로 8x1,024=8,192개의 Thread를 동시에 실행할 수 있다.

〈Table 2〉 Specification of graphic card

General Information	
Name	GeForce GTX 560 Ti
Compute Capability	2.1
CUDA Cores	384
Graphics Clock	882 MHz
Processor Clock	1645 MHz
Memory Information	
Memory Bandwidth	128 GB/sec
Total Global Memory	1,073,741,824
Total Constant Memory	65,536
Multiprocessor Count	8
Shared Memory per mp	49,152
Registers per mp	32768
Max Threads per Block	1024
Max Thread Dimensions	<1024, 1024, 64>
Max Grid Dimensions	<65535, 65535, 65535>

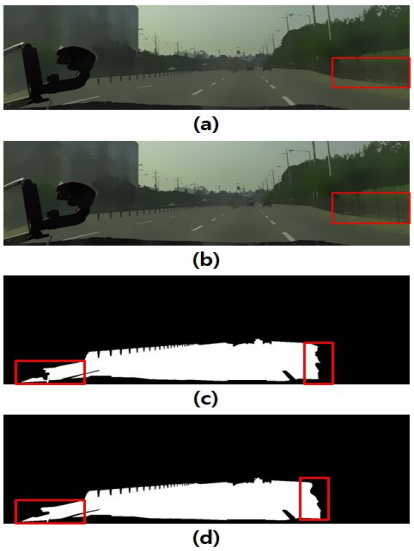
[Fig. 6]은 GPU 기반의 병렬 처리 방식을 나타낸다. 아래의 이미지 해상도는 1440x405 크기를 가지며 gpu::Mat 구조로 디바이스(GPU, Graphic Card) 내에서 병렬로 처리된다. OpenCV CUDA 함수를 사용하면 Mean-Shift, Split, Morphology, And operation에 대하여 Thread, Block, Grid 등을 최적의 상태로 할당하여 병렬 처리를 실행해줌으로써 최상의 처리 속도를 구현할 수 있다.



[Fig. 6] GPU based parallel processing method

4. 실험결과

[Fig. 7]은 OpenCV CPU와 OpenCV CUDA에 대한 시뮬레이션 결과를 비교한 그림이다. (b)가 (a)보다 컬러 분할이 잘 되었다. 이는 Host(PC)에서 보다 Device (Graphic Card)에서 정밀한 Mean-Shift 알고리즘이 수행되었음을 의미한다. 그리고 (d)는 (c) 보다 상대적으로 도로영역으로 인식하는 범위가 넓음을 확인할 수 있다.



[Fig. 7] Comparison of Simulation result
(a. Mean-Shift on Host b. Mean-Shift on Device c. Road Region Extraction on Hostd. Road Region Extraction on OpenCV CUDA)

본 논문에서의 가장 중요한 부분은 OpenCV CUDA를 활용하여 기존 알고리즘 대비 빠른 처리 속도를 가져왔느냐는 것인데, 이는 <Table 3>에 기존 알고리즘[4]과 제안하는 알고리즘의 처리 속도를 비교하였다. 또한 <Table 3>의 처리 속도 비교는 Host(PC)와 Device (Graphic Card) 사이의 데이터 전송 속도을 포함한다.

Mean-Shift Segmentation을 수행했을 경우, 기존의 CPU에서 순차적으로 처리할 때보다 GPU에서 병렬로 처리했을 경우 약 4.37배 정도의 빠른 처리 속도를 가져오는 것을 확인할 수 있다. 반면 Noise Removal and Road Extraction은 GPU의 병렬 처리 방식이 CPU의 직렬 처리 방식보다 느린 처리 속도 결과를 가져왔다. CPU에서의 처리속도가 GPU의 처리속도보다 빠른 이유에 대해서는 <Table 4>에 나타낸다.

<Table 4>는 알고리즘 처리 속도와 PC와 그래픽 카드 사이의 데이터 전송 속도를 비교한 표이다. <Table 4>를 통해서 그래픽 카드 내부에서 알고리즘 처리 속도는 병렬로 빠르게 처리하는 반면 PC와 그래픽카드 사이의 데이터 전송 과정에서 시간이 상대적으로 많이 소요됨을 확인할 수 있다.

〈Table 3〉 Processing speed comparison

	Conventional methods	Proposed methods
Mean-Shift Segmentation	5.50 sec	1.26 sec
Noise Removal and Road Extraction	0.02 sec	0.36 sec
Total	5.90 sec	1.91 sec

〈Table 4〉 Comparison of algorithm speed and data transmit speed on OpenCV CUDA

	Only Algorithm	Algorithm & Data transmit	Only Data transmit
Mean-Shift Segmentation	1.26 sec	0.43 sec	0.83 sec
Noise Removal and Road Extraction	0.36 sec	0.02 sec	0.34 sec

5. 결론

본 논문은 기존의 도로영역검출 알고리즘에 OpenCV CUDA 기반의 병렬 처리 방식을 적용함으로써 보다 향상된 처리 속도 결과를 목표로 한다. OpenCV CUDA는 CUDA C 언어에 비해 많은 함수들을 지원하므로 사용자로 하여금 빠른 결과 도출 및 분석의 편의성을 제공한다는 장점이 있다. 따라서 본 논문의 시뮬레이션은 OpenCV 2.4.6에 CUDA ToolKit 5.5와 연동하고 GeForce GTX 560 Ti 모델의 그래픽 카드를 사용하여 최종적으로 결과를 도출하였다.

OpenCV CUDA를 활용한 병렬 처리 방식의 제안하는 알고리즘은 Mean-Shift Segmentation, Noise Removal and Road Extraction 영역을 GPU기반의 병렬 처리를 수행함으로써 기존 알고리즘의 처리 속도 대비 약 3.09배의 처리 속도 향상을 가져온 것을 확인할 수 있었다.

반면 GPU 기반의 병렬 처리 방식은 빠른 처리 속도를

지원하지만, Host(PC)와 Device(Graphic Card)간의 데이터 전송에서 많은 시간을 소요되는 단점이 있어 앞으론 이런 문제점을 보완책 및 해결책에 대한 연구가 진행되어야 한다. 또한 실제 자동차에 접목시키기 위해서는 정확한 검출 알고리즘 연구를 통해 안전성·신뢰성을 확보할 필요성이 있다.

REFERENCES

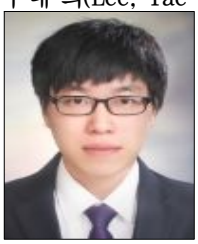
[1] Yongjin Yeom, Yongkuk Cho, "High-Speed Implementations of Block Ciphers on Graphics Processing Units Using CUDA Library", Journal of The Korea Institute of Information Security and Cryptology, vol. 18, no. 3, pp. 23-32, 2008.

[2] Jun-Chul Kim, Young-Han Jung, Eun-Soo Park, Xuenan Chui, Hak-il Kim, Uk-Youl Huh, "The Implementation of Fast Object Recognition Using Parallel Processing on CPU and GPU", Journal of Institute of Control, Robotics and System, vol. 15, no. 5, pp. 488-495, 2009.

[3] Kyoung-Hwan Park, Chi-Won Lee, Chang-Woo Lee, "Road Detection using Mean Shift Algorithm and Similarity Region Merging method", Workshop presentatio file, Korea Information Science Society, vol. 36, no.4, pp.437-440, 2009.

[4] Tae-Hee Lee, Bo-Hyun Hwang, Jong-Ho Yun, Byoung-Soo Park, Myung-Ryul Choi, "A Road Extraction Algorithm using Mean - Shift Segmentation and Connected-Component", Journal of Digital Convergence, Vol. 12, no1, pp. 359-364, 2014, 1

이 태 희(Lee, Tae Hee)



- 2005년 3월 : 인제대학교 나노공학부 (학사)
- 2012년 3월 : 한양대학교 전자통신공학과(석사)
- 관심분야 : SoC/ASICs 설계, Image Enhancement, Stereo Vision, 병렬처리
- E-Mai: hy12504684@hanyang.ac.kr

황 보 현(Hwang, Bo Hyun)

- 2004년 2월 : 한양대학교 전자컴퓨터공학과(학사)
- 2006년 2월 : 한양대학교 전자전기제어계측공학과 (석사)
- 2006년 1월 ~ 2007년 7월 : 동부하이텍 반도체부문 사원
- 2007년 9월 ~ 현재 : 한양대학교 전자전기계측공학과 박사과정
- 관심분야 : Image Processing, SoC/ASIC 설계, FPD Controller 설계, 2D/3D 영상처리
- E-Mail : jokersir@ymail.com

윤 종 호(Yun, Jong Ho)

- 2001년 2월 : 한양대학교 전자컴퓨터공학부 졸업(학사)
- 2003년 2월 : 한양대학교 전자전기제어계측공학과 졸업(석사)
- 2003년 2월 ~ 현재 : 한양대학교 전자통신전파공학과 박사과정
- 관심분야 : 영상처리, 2D/3D Image Processing, SoC/ASICs 설계
- E-Mail : sfw1179@hanmail.net

최 명 렬(Choi, Myung Ryul)

- 1983년 2월 : 한양대학교 전자공학과(학사)
- 1985년 2월 : 미시간 주립대학교 컴퓨터공학과(석사)
- 1991년 2월 : 미시간 주립대학교 컴퓨터공학과 (박사)
- 1991년 3월 ~ 10월 : 생산기술연구원 전자정보 실용화센터 조교수
- 1991년 11월 ~ 1992년 8월 : 생산기술연구원 산하 전자부품종합기술연구소 선임연구원
- 1992년 9월 ~ 현재 : 한양대학교 전자통신공학과 교수
- 관심분야 : SoC/ASICs 설계, FPD Controller 설계, 2D/3D 영상처리, 스마트카드/RFID 응용, ITS/EFC 응용
- E-Mail : choimy@hanyang.ac.kr