ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 13-03-22

Journal of IKEEE.Vol.17,No.3,371~377,September 2013
http://dx.doi.org/10.7471/ikeee.2013.17.3.371

병렬구조를 이용한 증강현실 구현

Implementation of augmented reality using parallel structure

박 태 룡 *, 허 훈*, 곽 재 창*★

Tae-Ryong Park*, Hoon Heo*, Jae-Chang Kwak*★

Abstract

This thesis propose an efficient parallel structure method for implementing a FAST and BRIEF algorithm based Augmented Reality. SURF algorithm that is well known in the object recognition algorithms is robust in object recognition. However, there is a disadvantage for real time operation because, SURF implementation requires a lot of computation. Therefore, we used a FAST and BRIEF algorithm for object recognition, and we improved Conventional Parallel Structure based on OpenMP Library. As a result, it achieves a 70%~100% improvement in execution time on the embedded system.

요 약

본 논문에서는 FAST와 BRIEF 알고리즘을 기반으로 하는 증강현실을 구현하기 위해서 효율적인 병렬 구조를 제안한다. 객체 인식 알고리즘으로 잘 알려진 SURF 알고리즘은 객체인식에 강인하지만 연산 량이 많아 실시간으로 구현하기에 어려운 단점을 가지고 있다. FAST와 BRIEF 알고리즘을 활용하여 객체를 인식하였고, 임베디드 환경에서 성능을 향상하기 위해 기존의 OpenMP 라이브러리를 사용한 병렬구조를 개선하여 속도를 약 70%에서 100%로 향상 시켰다.

Key words : SURF, FAST Corner Detection, BRIEF, Hamming Distance, OpenMP, Augmented Reality

Ⅰ. 서론

컴퓨터 비전 중 하나인 물체인식 기법은 카메라 또는 각종 센서 정보를 활용하여 일정 범위에서 물체의 존재 여부를 판단하는 기술로 많은 연구가 진행되고 있다.

* Dept. of Computer Engineering, Seokyeong University, trpark@skuniv.ac.kr. 02-940-7742

*★Dept. of Computer Science, Seokyeong University, jckwak@skuniv.ac.kr, 02-940-7758

★ Corresponding author

Manuscript received Ago. 30, 2013; revised Sep. 17, 2013; accepted Sep 23. 2013

많은 연구 중 최근 스마트 폰과 같은 임베디드 환경에서 인식 알고리즘 연구가 활발히 이루어지고 있다.

대표적인 예로는 증강현실이 있다. 증강현실에는 마커 기반의 증강현실과 비 마커 기반의 증강현실로 구분 된다. 비 마커 기반의 증강현실 알고리즘에서 사용되는 대표적인 알고리즘은 SURF[1]가 있다. 하지만 실시간으로 구현하기에는 연산 량이 많아 구현하기 어려운 단점을 가지고 있다. 이러한 단점을 보완하기 위해 연산 량이 적은 알고리즘이 필요하다. 특징점 기반 인식 알고리즘은 특징점 검출 단계와 표현자 생성 단계로 구성되어 진다. 특징점 검출 단계에서는 연산 량이 적은 알고리즘인 FAST Corner Detection 알고리즘[2]을 사용하고, 표현자 생성 단계에서는 BRIEF 알고리즘[3]을 사용해서 연산 량이 적

은 특징점 기반 인식 알고리즘을 구현 하였다. 하지만 FAST Corner Detection과 BRIEF 알고리즘만으로는 임베디드 환경에서 실시간으로 처리하기 어려운 단점이 있다. 그렇기 때문에 효율적인 병렬처리와 최적화 단계가 필요하다. 기존에는 OpenMP 라이브러리[4]를 사용한 병렬구조로 1.7배의 성능을 향상시켰지만, 개선된 병렬구조로 설계하여 2배의 성능으로 향상시켰다.

검증 환경으로는 ETRI에서 개발한 Snake Dual Core가 장착된 ALDEBARAN 플랫폼과 Exynos4421 Cortex-A9 Quad Core 장착된 ODROID-X 플랫폼에 구현하여 비교하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 병렬 기법을 이용한 물체인식 알고리즘을 소개하고, 3장에서는 개선된 병렬 기법을 적용한 물체인식 알고리즘을 소개한다. 4장에서는 실험 결과 및 분석으로 개선된 수행성능을 보여 준다.

II. 기존 병렬 기법을 이용한 물체인식 알고리즘

1. 물체 인식 알고리즘 기법

물체 인식 알고리즘 기법으로는 FAST Corner Detection과 BRIEF 알고리즘을 사용하였다. 그림 1과 같이 특징점 검출 단계인 FAST Corner Detection, 표현자 생성단계인 BRIEF 알고리즘 그리고, 매칭 단계인 Hamming Distance 알고리즘[5]이 수행 되어 진다.

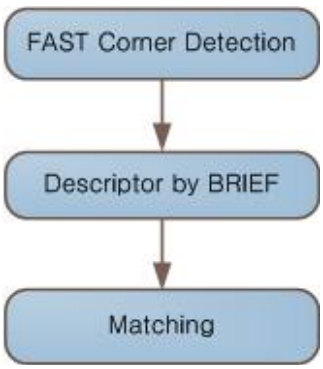

Fig. 1. Flow for FAST&BRIEF algorithm
그림 1. FAST&BRIEF 알고리즘 흐름

FAST Corner Detection 단계에서는 코너부분과 Blob영역을 특징점으로 검출하게 된다. Blob영역이란 밝은 부분에서 갑자기 어둡게 되는 부분을 뜻한다.

검출된 특징점마다 표현자를 생성해주는 단계가 표현자 생성 단계인 BRIEF이다. BRIEF 알고리즘은 비트열로 표현자를 생성한다. 즉, 검출된 각 특징점 중심으로 31x31영역을 잡고, 그 주변으로 임의로 2개의 좌표를 정해 밝기 값을 비교하여 1과 0인 값을 설정해 준다. 이러한 방식을 총 256번 반복해 256bit인 표현자를 생성하게 된다. 이 단계까지 수행이 되면 각 특징점의 위치와 256bit의 표현자의 데이터가 저장이 된다.

매칭 단계에서는 Hamming Distance 알고리즘을 사용한다. Hamming Distance는 bit 정보를 비교하는 것으로 xor 연산을 사용하여 총 틀린 bit 만큼 거리로 정한다. 예로 8bit인 정보가 2개가 있을 경우, 하나는 0010_1100이고 다른 하나는 0010_0111일 경우 xor 할 경우, 0000_1011이 된다. 그럼 '1'이 3개이므로 거리는 3이 된다. Hamming Distance 알고리즘을 사용하여 입력영상의 표현자 정보와 DB 영상의 표현자 정보를 비교하여 비슷한 정보가 있는지 찾게 된다. 특징점이 많이 검출 될수록 검사할게 많아져 매칭 단계에서 속도가 느려지는 문제점이 있다.

이 점을 보완하기 위해서 정렬 알고리즘[6]을 사용하여 각 검출된 특징점의 256bit 표현자를 적은 값부터 큰 값 순서대로 정렬을 한다. 그 후, 정렬된 표현자 정보를 가지고, 그 주변의 표현자 정보와 비교를 하게 되면 매칭 성능이 향상이 된다.

2. 기존 병렬 기법

기존 병렬 기법으로는 OpenMP 라이브러리를 사용하여 FAST Corner Detection과 BRIEF 알고리즘 그리고, 매칭 알고리즘을 병렬처리 하였다.

그림 2는 기존 병렬 구조로 FAST와 BRIEF 알고리즘 그리고, 매칭 알고리즘을 사용하여 병렬 처리 하는 과정을 보여준다.

기존의 병렬 구조는 해상도 640x480인 입력영상을 반으로 나눠 FAST알고리즘을 수행하여 Core_0은 0번 픽셀에서 640*240번 픽셀 데이터를 가지고, 특징점을 검출하게 되고, Core_1은 640*241번 픽셀부터 640*480번 픽셀 데이터를 가지고, 특징점을 검출하게 된다. 검출된 위치는 임의의 메모리에 저장해 놓았다가 검출된 특징점을 반으로 나눠 Core_0과 Core_1에 할당하여 BRIEF 알고리즘을 각각 수행한다. 그 후, 매칭 단계에서 검출된 특징점 위치와 표현자 정보를 반으로 나눠 일치하는 부분이 있는지 검사한 후, 일치하는 위치를 출력하게 된다.

그림 2와 같은 구조로 구현되면 속도는 향상되긴 하지만, 반복문인 과정에서만 병렬화가 되어 코어 수에 맞게 성능이 향상 되지 않는다.

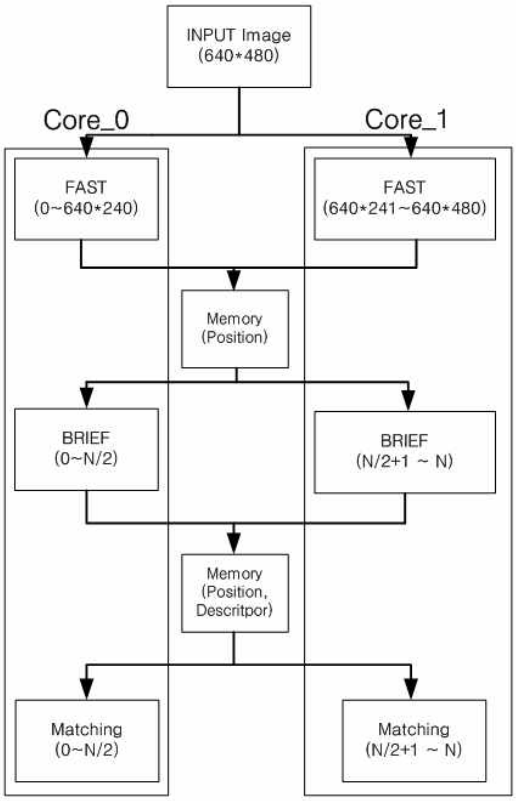

Fig. 2. Conventional Parallel Structure
그림 2. 기존의 병렬구조

III. 제안하는 병렬 구조

1. 제안하는 병렬 기법과 최적화

FAST와 BRIEF알고리즘을 수행하기 전에 전처리를 수행해야 된다. 전처리 과정으로는 Resize와 가우시안 필터 작업이 필요하다. Resize를 하는 이유는 크기가 변동해도 같은 물체인지 판별이 가능해지고, 가우시안 필터[8] 작업은 영상이미지의 노이즈를 제거해서 노이즈가 특징점으로 검출되는 것을 방지한다. 이와 같이 전처리 과정도 있기 때문에 그림 3과 같이 입력영상을 코어 별로 할당하여, 알고리즘을 수행을 하게 되면, 매칭 알고리즘에서 수행되어 나온 정보를 가지고, 결과 화면만을 출력하기 때문에 성능이 2배로 향상 되어 진다.

즉, Core_0과 Core1에서 각각 한 프레임씩 FAST & BRIEF 알고리즘과 Matching 알고리즘을 수행하기 때문에 성능이 2배로 향상 된다.

예를 들어 한 프레임에 0.05s가 걸린다면, 그림 3과 같은 병렬 처리를 수행을 하면, 두 프레임이 0.05s에 객체인식 알고리즘이 수행 되는 것이다. 이와 같은 방법이 설립되는 이유는 특징점 기반 인식 알고리즘을 수행하면, 카메라가 1초에 30프레임을 처리 한다 해도 객체인식 알고리즘 과정을 거치게 되면, 카메라에서 갖고 오는 속도가 느려지게 된다. 그 부분을 두 프레임 씩 데이터를 읽어 와 그림 3과 같은 병렬과정을 거쳐 수행하게 되면, 수행 시간을 향상 시킬 수가 있다.

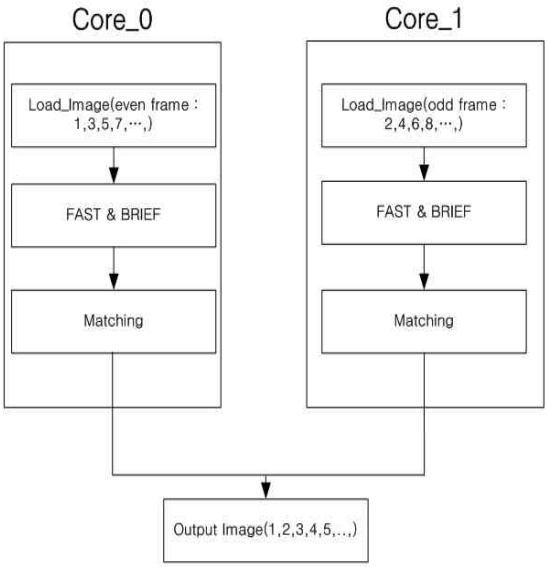

Fig. 3. Proposed Parallel Structure
그림 3. 제안하는 병렬 구조

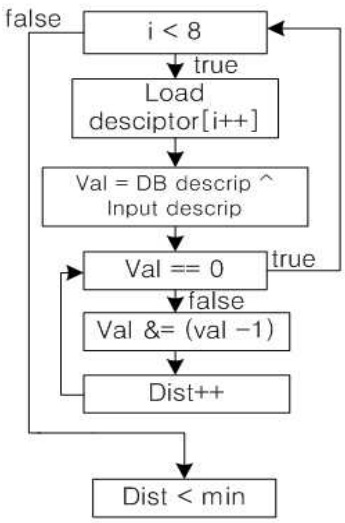

Fig. 4. Compute the distance of two integers using the Conventional Hamming Distance
그림 4. 기존의 Hamming Distance을 이용한 두 정수형의 거리 값 계산.

그림 4는 매칭 알고리즘인 Hamming Distance 알고리즘을 나타낸다. 그림 4의 순서도를 설명하면, 우선 표현자는 256비트로 구성되어 있다. 정수형이 4byte이기 때문에 8개의 정수형이 필요하다. 입력영상의 256bit로 구성된 표현자 정보를 8개의 정수형으로 읽어온 후, DB 영상의 표현자 정보와 xor 연산을 수행하여 xor 연산된 값이 0이 될 때 까지 거리 값을 증가한다. 거리 값을 가지고, DB영상의 표현자 정보와 Input 영상의 표현자 정보가 같은지 판별 할 수가 있다. 256bit를 다 비교한 후 거리 값이 임계치 값 보다 작으면 일치한다고 판별하게 된다.

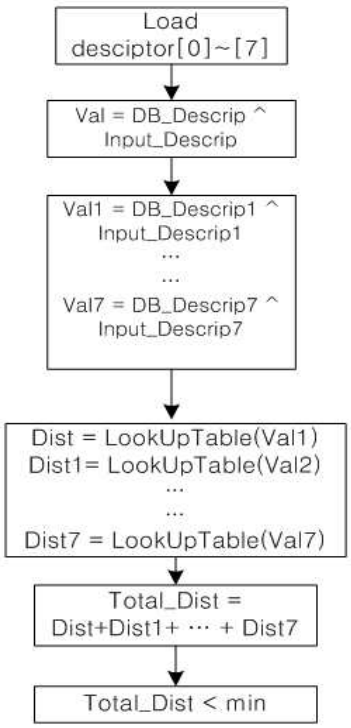

Fig. 5. Compute the distance of two integers using the Proposed Hamming Distance.
그림 5. 제안하는 Hamming Distance을 이용한 두 정수형의 거리 값 계산.

2. 호모그래피 행렬을 이용한 증강 현실

매칭 된 특징점 수를 RANSAC 기법을 이용하여 호모그래피 행렬을 구한 후, 호모그래피 행렬을 가지고, 3D 물체나 동영상을 물체인식이 된 좌표에 출력을 한다.

호모그래피 행렬을 구하는 식[9]은 다음과 같다.

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} H_{11} H_{12} H_{13} \\ H_{21} H_{22} H_{23} \\ H_{31} H_{32} H_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \Leftrightarrow X_2 = HX_1 \quad (1)$$

수식 1에서 H는 호모그래피 행렬 3x3을 뜻하고, $x1,y1,x2,y2$는 두 영상 화면에 투영되는 점을 뜻한다.

H 값을 구하기 위해서 수식 1을 수식 2와 3 같이 표현 한다.

$$x_2' = \frac{H_{11}x_1 + H_{12}y_1 + H_{13}z_1}{H_{31}x_1 + H_{32}y_1 + H_{33}z_1} \quad (2)$$

$$y_2' = \frac{H_{21}x_1 + H_{22}y_1 + H_{23}z_1}{H_{31}x_1 + H_{32}y_1 + H_{33}z_1} \quad (3)$$

다음은 수식 2와 수식 3의 분모를 수식 2와 수식 3 양쪽에 곱하고, 평면상에서 H 값을 구하기 때문에 z 축은 1로 설정해 준다. 그 후 식을 재정리하면 수식 4와 같이 나오게 된다.

$$\begin{aligned} a_x h &= 0 \\ a_y h &= 0 \end{aligned} \quad (4)$$

여기서 ax, ay, h는 수식 5와 같다.

$$\begin{aligned} h &= (H_{11}, H_{12}, H_{13}, H_{21}, H_{22}, H_{23}, H_{31}, H_{32}, H_{33})^T \\ a_x &= (-x_1, -y_1, -1, 0, 0, 0, x_2'x_1, x'^2y_1, x'^2) \\ a_y &= (0, 0, 0, -x_1, -y_1, -1, y'^2x_1, y'^2y_1, y'^2) \end{aligned} \quad (5)$$

두 영상에서 4개 이상의 n개의 대응점이 주어지면, 수식 6과 같은 시스템을 구축할 수가 있다.[10]

$$Ah = 0, A = \begin{pmatrix} a_{x1}^T \\ a_{y1}^T \\ \vdots \\ a_{xN}^T \\ a_{yN}^T \end{pmatrix} \quad (6)$$

수식 6에서 재차 선형 최소자승법을 사용하여 풀면, h 값을 구할 수가 있다.

Ⅳ. 실험 결과 및 분석

그림 6은 PC환경에서 FAST&BRIEF 알고리즘과 후처리로 OpenGL를 사용하여 증강현실을 구현한 결

과 화면이다. FAST&BRIEF 알고리즘과 매칭 알고리즘을 병렬 구조로 설계하여 실시간으로 증강현실 할 수 있도록 구현 하였다.

Fig. 6. Result Screens
그림 6. 결과 화면

표 1은 논문 [7]과 비교한 결과이다. 표 1은 Nexus-7 1.5GHz에서 SURF를 이용한 증강현실을 구현한 결과[7] 와 Cortex-A9 1.4GHz 에서 FAST & BRIEF 알고리즘을 이용한 증강현실 그리고, 인텔 I7 3.4GHz에서 OpenSURF 알고리즘 성능을 비교한 결과이다. [7]은 DB 영상의 크기는 400x300으로 한 프레임 당 0.55s가 걸리면서 매칭 된 특징점 개수는 73pair로 나와 있고, 본 논문은 DB 영상으로 인식에서 잘 사용하는 640x480 이미지인 꽃, 카드, 그래피티로 테스트 해 본 결과, 매칭 된 특징점 개수를 약 80 pair ~ 90 pair로 기준을 잡으면, DB 특징점 개수는 724, 550, 700이고, 카메라 영상의 특징점 개수는 약 400 ~ 500개가 추출이 되어, 프레임 당 0.11s ~ 0.13s 처리성능을 확인 할 수가 있었다. OpenSURF에서도 매칭 된 특징점 개수를 약 80 pair ~ 90 pair로 잡을 때, DB 특징점 개수는 698, 475, 653이고, 카메라 영상의 특징점 개수는 약 400 ~ 500가 추출이 되어, 프레임 당 0.21s ~ 0.27s 처리성능이 나왔다.

[7]은 (400x300)/(1500MHz x 550ms) = 0.00014 pixel/clock cycle의 성능을 보여주고, 본 논문은 (640x480)/(1400MHz x 120ms) = 0.0018 pixel/clock cycle의 성능을 보여준다.

Table 1. compare [7] and FAST & BRIEF performance
표 1. [7]과 FAST & BRIEF 성능 비교

	Exec. Time	Frequency	Pixel per clock cycle
SURF + Matching[7]	550ms	1.5GHz	0.00014
OpenSURF + Matching	230ms	3.4GHz	0.00039
FAST&BRIEF + Matching	120ms	1.4GHz	0.0018

표 2과 표 3은 300MHz로 동작하는 Snake 프로세서와 1.4GHz로 동작하는 Cortex-A9 CPU 에서 FAST&BRIEF 알고리즘 그리고, Matching 알고리즘까지 구현한 결과 수행성능을 보여 준다.

Cortex-A9에서는 기존 방식의 OpenMP 라이브러리를 사용한 병렬구조인 결과[6]를 보여주고, Snake 프로세서에서는 Snake용 병렬 라이브러리와 개선된 병렬구조를 사용한 결과를 보여준다.

Table 2. Object Recognition Algorithm Execution time implementing Single Core
표 2. 싱글 코어로 구현한 객체인식 알고리즘 성능 표.

Processor(MHz)	FAST& BRIEF	Matching	Total
Snake (300MHz)	0.18s	0.2s	0.38s
Cortex-A9[6] (1400MHz)	0.045s	0.066s	0.12s

Table 3. Object Recognition Algorithm Execution time implementing Dual Core
표 3. 듀얼 코어로 구현한 객체인식 알고리즘 성능 표.

Processor(MHz)	FAST& BRIEF	Matching	Total
Snake (300MHz)	0.09s	0.1s	0.19s
Cortex-A9[6] (1400MHz)	0.035s	0.036s	0.071s

표 2와 표 3을 비교해 보면 Snake 프로세서에서 제안하는 병렬구조로 구현한 결과 FAST&BRIEF 알고리즘은 0.18s에서 0.09s로, Matching 알고리즘은 0.2s에서 0.1s로, 성능이 2배 향상 되었다. 그리고 Cortex-A9 에서는 기존의 병렬구조로 구현한 결과 FAST&BRIEF 알고리즘은 0.045s 에서 0.035s로, Matching 알고리즘은 0.066s 에서 0.036s로, 성능이 약 1.7배 향상된 것을 확인 할 수가 있다.

Cortex-A9에서는 OpenMP 라이브러리를 사용하여 반복문 부분에서만 병렬 처리했기 때문에 약 1.7배 정도 성능이 향상 되었고, Snake Processor의 경우에는 Snake에서 지원하는 병렬 라이브러리와 제안하는 병렬 구조를 사용해서, 반복문 뿐 만 아니라 변수 선언부터 종료부분까지 병렬처리로 가능하기 때문에 성능이 2배로 좋아질 수 가 있었다.

V. 결 론

본 논문에서는 물체인식 알고리즘을 이용하여 증강현실을 구현하였고, 수행 성능을 더 향상시키기 위해서 효과적인 병렬구조로 설계하였다.

검증 환경으로는 임베디드 환경에서 SURF 알고리즘으로 구현한 증강현실과 FAST&BRIEF 알고리즘으로 구현한 증강현실 동작 성능에 대해서 비교 하였고, Cortex-A9 과 Snake 프로세서에서 기존의 병렬구조와 개선된 병렬 구조로 구현하여 약 1.7배 성능 향상에서 2배로 성능이 향상되는 것을 확인 하였다.

References

[1] H. Bay, E. Andreas, T. Tuytelaars and L. V. Gool, "Speeded-up roubst features", Computer Vision and Image Understanding, Vol 110, Issue 3, pp 346-359, June 2008

[2] E. Rosten and T. Drummond. "Machine learning for high-speed corner detection", In European Conference on Computer Vision, volume 1, 2006. 1

[3] M. Calonder, V. Lepetit, C. Strecha, and P .Fua, "Brief: Bi-nary robust independent elementary features". In European Conference on Computer Vision. 2010.

[4] OpenMP Architecture Revice Board. "OpenMP Apllication Program Interface", www.open-mp.org, version3.0, May 2008

[5] Hamming, Richard W. "Error detecting and error correcting codes", Bell System Technical Journal, pp 147-160, 1950.

[6] Park Tae Ryong, "Implementation of Real time based Multi-object recognition algorithm", Journal of IKEEE, pp 51-56, 2013. 3

[7] Ji-Yean Yoon, Il-Young Moon, "The Study on Marker-less Tracking Algorithm performance based on Mobile Augmented Reality", Journal of KONI, vol 16, No 6, 2012. 12

[8] R.A. Haddad and A.N Akansu, "A Class of Fast Gaussian Binomial Filters for Speech and Image Processing", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 39, pp 723-727, 1991.

[9] D. Kriegman, "Homography Estimation", Lecture Computer Vision, CSE A, 2007.

[10] Ki-Yong Lee and Joon-Woong Lee, "Ground Plane Detection Using Homography Matrix", Journal of Institute of Control, Robotics and Systems, pp 983-988 ,2011

BIOGRAPHY

Heo Hoon (Student Member)

2012: BS degree in Computer Engineering, Seokyeong University
Present:: MS course in Electronics Computer Engineering, Seokyeong University
<Research interests>
Microprocessor, Embedded System, Image Processing

Park Tae Ryong(member)

1985 : Hangyang University, Dept. of Mathmatics(BS)
1987 : Hangyang University, Dept. of Mathmatics(MS)
1995 : Hangyang University, Dept. of Mathmatics(Ph.D)
1994~: Seokyeong University, Dept. of Computer Engineering, Professor
<Research interests> Crypto Algorithm, Computer Security, Computer Arithmetic, Recognition Algorithm

Kwak Jae Chang(member)

1983 : Yonsei University, Dept. of Bachelor of Arts(BS)
1989 : Univ. of Iowa, Dept. of Computer Science(MS)
1993 : Univ. of Iowa, Dept. of Computer Science(Ph.D)
1995~: Seokyeong University, Dept. of Computer Science, Professor
<Research interests> Network Traffic Control, QoS, Realtime Scheduling, Embedded System