논문 2014-09-42

사람 인식을 위한 비 이미지 개선 및 고속화

(Raining Image Enhancement and Its Processing Acceleration for Better Human Detection)

박 민 웅, 정 근 용, 조 중 휘*
(Min-Woong Park, Geun-Yong Jeong, Joong-Hwee Cho)

Abstract : This paper presents pedestrian recognition to improve performance for vehicle safety system or surveillance system. Pedestrian detection method using HOG (Histograms of Oriented Gradients) has showed 90% recognition rate. But if someone takes a picture in the rain, the image may be distorted by rain streaks and recognition rate goes down by 62%. To solve this problem, we applied image decomposition method using MCA (Morphological Component Analysis). In this case, rain removal method improves recognition rate from 62% to 70%. However, it is difficult to apply conventional image decomposition method using MCA on vehicle safety system or surveillance system as conventional method is too slow for real-time system. To alleviate this issue, we propose a rain removal method by using low-pass filter and DCT (Discrete Cosine Transform). The DCT helps separate the image into rain components. The image is removed rain components by Butterworth filtering. Experimental results show that our method achieved 90% of recognition rate. In addition, the proposed method had accelerated processing time to 17.8ms which is acceptable for real-time system.

Keywords : Human detection, Rain removal, Image filtering, Morphological components analysis, Parallel processing, Image decomposition

Ⅰ. 서론

최근 들어 측정, 검사, 군사 등의 분야에서 자동화 시스템의 수요가 급증하면서 다양한 인식 기술이 요구되고 있다 [1-4]. 그 중에서 보행자 인식 시스템, 감시 시스템 등에서 사람 인식 알고리즘의 활용 빈도가 높기 때문에 다양한 방향에서 연구가 진행 중이다 [5-9].

현재 연구되고 있는 알고리즘들 중에서 윤곽선의 기울기 특성을 이용한 HOG-SVM(Support Vector Machine)이 많이 사용되고 있다 [5]. 그러나 그림 1처럼 비, 눈, 안개와 같은 외부 노이즈 성분에 의해 사람의 윤곽선이 가려질 경우 인식률이 저하된다. 그림 2는 인위적인 비 영상을 합성하여 HOG-SVM 알고리즘을 적용하였을 때 인식률이 줄어든 것을 보여준다. 이러한 문제를 해결하기 위해 노이즈가 있는 이미지를 개선하려는 연구가 진행되고 있다 [10-12]. 하지만 빗줄기 제거 알고리즘을 처리하는데 많은 시간이 소요되기 때문에 실시간 시스템에 적용하기 어렵다.

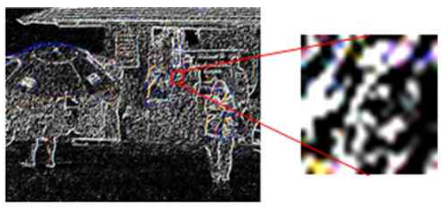

그림 1. 빗줄기에 의한 윤곽선의 왜곡
Fig. 1 Distorted edge image by rain streaks

본 논문에서는 빗줄기에 의해 인식률이 저하되는 문제를 해결하고 실시간 시스템에 적용 가능한

*Corresponding Author(jcho@incheon.ac.kr)
Received: 22 July 2014, Revised: 27 Aug. 2014, Accepted: 22 Sep. 2014.
M.W. Park, G.Y. Jeong, J.H. Cho: Incheon National University
※ 본 논문은 인천대학교 임베디드시스템공학과에서 수행하였음.

 2014 Dec.: 9(6) 345-351
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2014.9.6.345

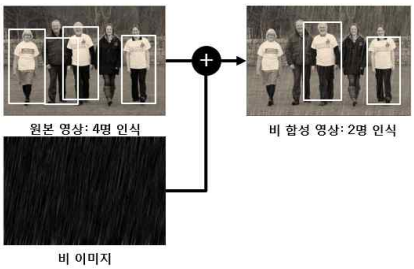

그림 2. 빗줄기로 인한 인식률 감소
Fig. 2 Recognition rate decrease by rain component

알고리즘을 제안하였다. 이 알고리즘을 적용함으로써 빗줄기 노이즈를 감소시키고 사람 인식률을 맑은 날 대비 90%의 수준으로 개선하였다. 또한, 자동차 응용 분야, 감시 분야 등에서 적용하기 위해 실시간으로 처리되도록 병렬 처리하였다.

Ⅱ. 관련 연구

비, 눈, 안개와 같은 기상상황은 물체 인식, 위치 추적 등 영상처리 시스템에 영향을 준다. 이러한 문제를 해결하기 위해 빗줄기를 제거하여 이미지를 개선하기 위한 연구가 진행 중이다.

빗줄기를 제거하기 위한 방법으로 연속적인 이미지의 차성분을 이용하는 방법과 시뮬레이션을 통해 빗줄기를 모델링하는 방법이 있다 [10, 11]. 그러나 많은 양의 데이터가 필요하고 예외조건이 많아서 인식률이 떨어지는 문제가 발생하였다.

이러한 문제를 해결하기 위해 MCA 기반으로 한 장의 이미지를 이용하여 빗줄기를 제거하는 방법이 연구되었다 [12]. MCA 기반의 알고리즘은 처음에 저주파 성분과 고주파 성분을 Bilateral 필터를 이용해 분리해낸다 [13]. 그리고 고주파 성분을 그림 3과 같이 몇 개의 조각으로 분리하고 조각 이미지를 이용해 사전학습(dictionary learning)을 시킨다 [14]. 학습된 데이터는 군집(K-means) 알고리즘으로 비 데이터와 비가 아닌 데이터로 분류하고 사전 데이터를 이용해 이미지에서 빗줄기를 분리하게 된다 [15]. 하지만 이와 같은 방법은 사전학습과 군집 알고리즘은 복잡한 연산이 사용되기 때문에 처리시간이 오래걸린다. MCA 기반의 알고리즘을 MATLAB으로 실행하였을 때 수행속도를 측정해 본 결과 124,000ms가 소요되었고 실시간 시스템에는 적용하기 어려운 것을 알 수 있다.

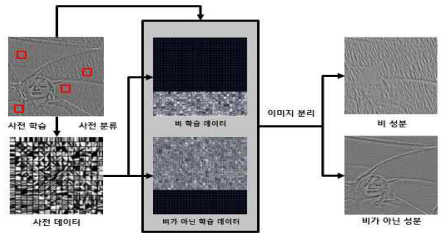

그림 3. MCA 기반 이미지 분리 과정
Fig. 3 Image decomposition based on MCA

그림 4. 비 제거를 위한 전처리 알고리즘의 추가
Fig. 4 Pre-processing addition for rain removal

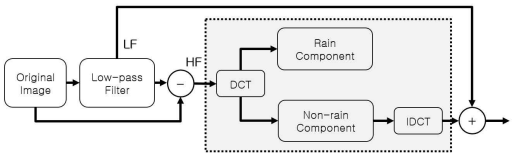

그림 5. 빗줄기 제거 알고리즘 구성
Fig. 5 Composition of rain removal algorithm

Ⅲ. 제안 방법

1. 빗줄기 제거 알고리즘 개요

악천후 이미지에서 사람의 인식률을 높이기 위해서는 사람에 해당하는 외곽선의 손실을 최소화하면서 비에 해당하는 성분만을 제거해야 한다. 비는 이미지 전반에 걸쳐 골고루 분포되어 있고 반복되는 패턴을 가지고 있어 고주파 성분과 유사한 특성을 가진다 [11]. 그러므로 악천후 이미지의 빗줄기는 그림 4처럼 고주파 성분을 제거하는 필터를 추가하여 비 성분을 제거한 후 사람 인식하는 과정을 거치게 된다.

빗줄기 제거는 그림 5처럼 2단계로 이루어진다. 1단계는 이미지에 섞여 있는 빗줄기 성분을 제거하기 위해 고주파 성분을 제거할 수 있는 저주파 통과 필터를 적용한다. 필터는 Bilateral 필터보다 수행속도가 빠른 Wiener 필터를 사용한다. 그러나 이미지의 고주파 성분에는 빗줄기뿐만 아니라 사람의 윤곽선 성분도 포함되기 때문에 저주파 통과 필터

그림 6. 빗줄기의 주파수 특성 분석
Fig. 6. Frequency analysis of rain

그림 7. 빗줄기 제거 필터 설계
Fig. 7 Filter design for rain removal

에 의해 사람 윤곽선까지 제거된다. 이렇게 제거된 사람의 윤곽선을 복원하기 위해 2단계에서는 주파수 도메인으로 변환한 후 빗줄기 특성을 분석하여 빗줄기만 제거하여 다시 빗줄기가 제거된 공간 도메인 성분을 획득한다. 그리고 1단계의 저주파 성분과 합쳐 복원하는 단계로 이루어진다.

2. 빗줄기 이미지 특징 분석

2단계에서 빗줄기만 제거하기 위해서는 빗줄기 특징을 분석해야 한다. 빗방울은 빛을 반사하기 때문에 흰색을 띠게 되고 불규칙하게 떨어지기 때문에 제거하기 어렵다. 그러나 빗방울은 셔터가 열리고 닫히는 순간 빠르게 움직이므로 획득한 이미지에서 빗방울은 항상 선형을 띈다. 그리고 빗방울은 지구의 중력에 의한 지면에 수직 아래 방향의 힘과 바람에 의한 지면에 수평 방향의 힘이 작용한다. 이러한 두 힘으로 빗방울은 비스듬하게 떨어지게 되고 지면을 기준으로 약 90°±45° 정도 기울어진 빗줄기 형태를 띠게 된다. 또한, 카메라의 셔터 속도가 일정하다면 이미지에서 빗줄기의 길이는 장마철 기준 빗방울의 두께가 3mm일 때, 빗방울의 종단속도가 8~9m/s 정도로 일정하다.

이런 특징 때문에 빗줄기는 일정 범위 내의 각도로 떨어지고 균등하게 떨어지게 된다. 이미지를 주파수 도메인으로 변환하였을 경우 빗줄기는 그림 6처럼 독특한 특성으로 나타난다. 즉, DCT을 할 경우 빗줄기 성분은 주파수가 0인 점을 기준으로 수평축에 주파수 성분이 모이기 때문에 이 부분에 주파수 성분을 제거하면 빗줄기를 제거할 수 있다. 공간 도메인에서 주파수 도메인으로 변환하는 방법으로 본 논문에서는 식(1)의 16x16 DCT을 사용한다.

$$F = CXC^T \tag{1}$$

$$C(k,n)\begin{cases}\dfrac{1}{\sqrt{N}}, & k=0 \\ & 0 \le n \le N-1' \\ \sqrt{\dfrac{2}{N}}\cos\left(\dfrac{\pi(2n+1)k}{2N}\right), & 1 \le k \le N-1 \\ & 0 \le n \le N-1'\end{cases}$$

DCT로 변환된 이미지에서 빗줄기 성분을 제거하기 위해서 저역통과필터를 사용한다. 주파수 도메인에서 Gaussian 필터를 사용하는 경우 링잉 현상이 일어나기 때문에 Butterworth 필터를 이용해서 그림 7처럼 빗줄기 성분이 있는 지역만을 식(2)와 같이 저역통과필터를 구성하여 빗줄기 성분을 제거한다.

$$H(x,y) = \frac{1}{1+(D(x,y)/D_0)^{2n}} \tag{2}$$

Ⅳ. 고속화 구현

1. GP-GPU 기반의 고속화 구현

본 논문에서 640x480 이미지의 각 픽셀을 병렬로 처리하는데 있어서 블록과 스레드를 각각 2차원으로 사용하였다. 블록 당 스레드 크기는 16x16으로 하였고 블록의 크기는 40x30으로 하였다. 또한, 2차원 영역에 대한 평균, 분산을 구하는데 있어서 이 중 반복문보다는 두 개의 반복문으로 나누어서 처리하는 것이 연산량이 적기 때문에 가로와 세로 두 단계로 나누어서 처리하였다. Wiener 필터의 처리는 그래픽 처리장치에서 캐시메모리에 해당하는 공유메모리를 이용해서 더 빠른 속도로 데이터 접근이 가능하다. 공유메모리는 각각의 블록 내에서만 접근할 수 있기 때문에 스레드의 크기인 16x16에서 좌우 상하로 +4 크기인 20x20의 공유 메모리에 이미지 데이터를 복사한다.

9x9 영역에서 구한 분산들의 전체 평균을 구하기 위해서 640x480개의 픽셀의 합을 구할 경우, 이를 순차적인 방법으로 계산하기에는 매우 오랜 시간이 걸리기 때문에 그림 8의 병렬 덧셈 축소(reduction) 방법을 이용하였다 [16]. 그래픽 처리장치의 공유 메모리를 가장 효율적으로 사용할 수 있도록 최적화 구현하였고 병렬로 처리하기 때문에 연산량이 감소하는 효과를 얻을 수 있다.

이와 같이 병렬로 처리된 데이터를 이용해 Wiener 필터를 구현할 수 있다. Wiener 필터를 적용한 후에는 제거된 고주파 성분을 추출하기 위해서 원본이미지와 Wiener 필터를 적용한 이미지의

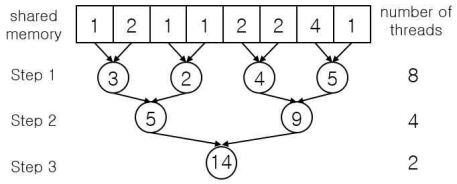

그림 8. 병렬 리덕션
Fig. 8 Parallel reduction

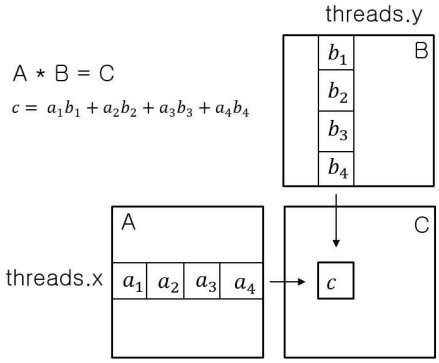

그림 9. CUDA을 이용한 행렬 곱 형태
Fig. 9 CUDA matrix multiplication

차 성분을 추출하게 된다. 다음으로 DCT을 하기 전에 밝기 성분만 추출하기 위해 RGB 모델에서 YCbCr 모델로 변환하는 작업을 수행한다.

DCT와 IDCT (Inverse Discrete Cosine Transform)의 경우 각 블록에서 코사인 변환 행렬과의 행렬 곱 연산을 통해 그림 9처럼 병렬 처리될 수 있다. 모든 데이터는 공유메모리에서 처리되고 DCT를 통해 주파수 도메인으로 변환한 후 각 스레드에서 16x16 비 제거 주파수 필터를 적용해 비 성분을 제거하고 IDCT를 통해 다시 공간 도메인으로 변환한다.

마지막으로 처리된 이미지를 다시 RGB 모델로 변환한 후 Wiener 필터 이미지와 더해서 비가 제거된 이미지를 획득하게 된다.

2. CPU 기반의 고속화 구현

Intel CPU에서 사용가능한 IPP(Integrated Performance Primitives) 라이브러리는 병렬화가 가능한 알고리즘에 대해서 최적화된 코드들을 제공해줌으로써 개발자가 원시적인 함수들을 개발하는데 걸리는 되는 시간을 절감시켜준다 [17]. 영상처리의 경우에는 필터링, 이미지 통계, 산술/논리 연

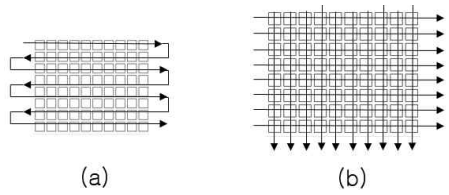

그림 10. 순차처리(a)와 병렬처리(b) 차이
Fig. 10 Difference between serial and parallel

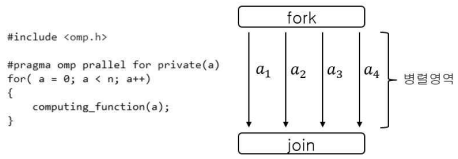

그림 11. OpenMP 이용한 멀티 스레드 수행
Fig. 11 Multi-thread process using OpenMP

산, 주파수 변환(DCT, FFT), 이미지 크기 변경, 어파인 변환 등을 최적화하여 제공해준다. 내부에서 벡터 메모리와 멀티코어를 이용하기 때문에 그림 10처럼 한 번에 여러 개의 데이터를 처리하여 속도를 향상한다.

본 논문에서는 IPP 라이브러리를 이용하여 Wiener 필터와 DCT을 고속화하여 수행하였다. 나머지 반복문에서는 OpenMP를 이용한 멀티쓰레드 방법을 사용하였다.

OpenMP는 병렬 처리 프로그램을 구현하는데 가장 간단히 사용할 수 있는 라이브러리로 그림 11처럼 '#pragma'을 사용하여 멀티 스레드로 실행되는 병렬 영역을 정의한다. 그러나 병렬 제어 지시어만으로는 멀티 스레드가 생성될 뿐, 병렬 영역의 작업 분할이 원활하게 이루어지지 않아 프로그램의 효율이 향상되지 않는다. 프로그램의 수행을 위해서 병렬 제어 지시어와 함께 스레드 별로 작업 분할(Work Sharing)을 지정해야 증가한 스레드 수만큼 프로그램의 효율이 향상된다. 또한, 병렬 제어 지시어 영역의 안과 밖에 있는 데이터를 명시하여 데이터를 동기화해야 한다.

V. 실험 및 고찰

1. 구현 환경

컴퓨터의 사양은 Intel i7-3770으로 연산 체계 64bit, 동작 클록 3.5GHz이다. 그래픽 처리장치의

표 1. 전처리 알고리즘 적용한 사람 검출 수
Table 1. Number of human detections in pre-processing algorithm

	전체 사람 수	검출 수			정확히 검출된 사람 수		
		원본 [5]	MCA [12]	제안	원본 [5]	MCA [12]	제안
image 1	6	3	5	6	3	5	6
image 2	4	1	2	3	1	2	3
image 3	4	5	2	4	4	2	4
image 4	8	7	8	8	7	8	8
image 5	6	4	2	4	4	2	4
image 6	2	0	1	3	0	0	2
image 7	6	3	5	5	3	5	5
image 8	5	3	5	5	3	4	5
image 9	4	2	1	3	2	1	3
image 10	5	3	4	4	3	4	4
합계	50	31	35	45	30	33	44

성능은 NVIDIA사 GTX670으로 CUDA 코어 1344개, 915Hz이고 공유 메모리와 스레드 개수는 최적화하여 사용하였다.

2. 제안 알고리즘 적용에 따른 인식률

MCA을 이용한 빗줄기 제거 방법의 인식률 문제와 연산량 문제를 해결하기 위해 DCT 기반의 빗줄기 제거 방법을 제안하였다. 연산량 측면에서 MATLAB 기반에서 MCA을 이용한 빗줄기 제거 방법을 수행하였을 때 약 124,000ms의 처리시간이 걸렸고 DCT 기반의 빗줄기 제거 방법을 수행하였을 때 약 882ms의 처리시간이 걸렸다. MCA을 사용했을 때 보다 140배 정도로 처리속도가 향상되었다는 것을 알 수 있다.

인식률은 검출률(recall rate)과 정확도(precision) 기준으로 비교하였다. 검출률은 입력 이미지의 전체 사람 중에서 성공적으로 검출해낸 사람의 비율이고 정확도는 검출된 결과 중 정검출(true positive)의 비율이다 [18]. 그림 12과 같이 총 10개의 이미지에 대해서 인식률을 비교하였고 결과는 표 1과 같다. 검출률은 비가 오는 이미지를 그대로 HOG-SVM으로 처리 할 경우 62%, MCA을 적용한 이미지의 경우 70%, 제안한 DCT을 적용한 이미지의 경우 90%로 MCA 방법보다 20%의 성능 향상을 보였다. 정확도도 제안한 방법이 다른 방법들 보다 약 20% 높은 결과를 얻을 수 있었다.

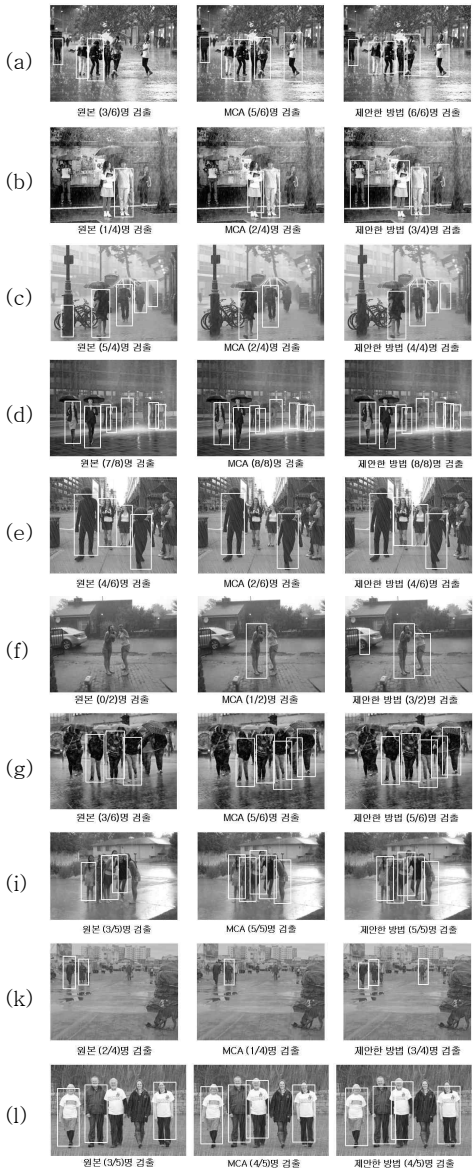

그림 12. 알고리즘에 따른 검출 수 비교
Fig. 12 Comparison of the recall rate

3. 병렬처리 수행시간 성능

HOG-SVM 알고리즘을 처리하는데 표 2에서 최대 18.7ms까지 처리속도를 고속화하였다. 그러나 제안한 전처리 알고리즘의 수행시간이 574.4ms이고 HOG-SVM 알고리즘 시간과 더하게 되면 총 처리시간이 593.1ms가 되어 실시간 시스템에 적용하기 어려워진다. 실시간 시스템에 적용하기 위해서 제안한 전처리 알고리즘을 CPU, GPU 기반으로 병

표 2. HOG-SVM 알고리즘 처리시간 (단위 : ms)
Table 2. Processing time of HOG-SVM

	CPU	GPU
HOG-SVM	1,250.0	18.7

표 3. 제안한 알고리즘 고속화 처리시간 (단위 : ms)
Table 3. Accelerated processing time of proposed algorithm

	Serialized	IPP	CUDA
Wiener Filter	507.0	2.0	1.2
DCT Filter	52.7	6.8	1.7
Others	14.7		
Total	574.4	23.5	17.6

렬처리 수행하였다.

제안한 알고리즘의 총 처리 시간은 표 3에서 Intel의 IPP 라이브러리를 사용하여 처리하였을 경우 23.5ms로 약 24배 처리속도가 향상되었고 NVIDIA의 CUDA를 이용하여 병렬 처리하였을 경우 14.8ms로 약 32배 처리속도가 향상되었다. GPU를 이용한 병렬처리의 경우 대용량 데이터에 대해 복잡한 연산을 처리할 경우에 병렬 처리 효과가 극대화되고 제안한 알고리즘의 Wiener 필터, DCT 등이 병렬화에 적절한 알고리즘이기 때문에 CUDA를 이용한 고속화 처리의 경우 총 처리 시간이 가장 적게 나왔다.

Ⅵ. 결론

본 논문은 빗줄기에 의해 저하된 사람 인식률을 개선하기 위한 빗줄기 제거 알고리즘을 제안하고 실시간 적용을 위해 병렬 처리 구현하였다. 주파수 도메인에서 빗줄기의 특성을 분석하고 빗줄기 특성을 제거할 수 있는 필터를 설계하여 인식률을 향상했고 MCA 기반의 알고리즘보다 적은 연산량을 가지는 알고리즘을 대체 적용하여 처리시간을 줄였다. 또한, CUDA와 IPP 라이브러리를 이용하여 병렬처리 구현하여 실시간 처리할 수 있도록 고속화 구현하였다.

빗줄기에 해당하는 노이즈뿐만 아니라 안개, 눈, 우박과 같은 기상상황 또는 조도의 불균형에 의해 사람 인식률이 저하되기 때문에 이를 개선할 수 있는 알고리즘이 요구된다. 이를 위해 센서를 이용해 기상상황을 파악한 후 기상상황에 맞는 알고리즘을 대체 적용할 수 있는 시스템을 구축해야 할 것이고 비나 눈이 오는 날에 우산, 우비 등과 같은 예외상황이 발생하기 때문에 기상상황에 맞게 예외상황을 판별할 수 있는 알고리즘을 개발해야 할 것이다. 또한, 프로세서에 따라 제안한 병렬 처리 방법이 지원되지 않기 때문에 다양한 고속화 방안을 모색하여야 할 것이다.

References

[1] H.K. Kim, "Lane detection for adaptive control of autonomous vehicle," IEMEK J. Embed. Sys. Appl. Vol. 4, No. 4, pp. 180-189, 2009 (in Korean).

[2] W.C. Jang, "Design and implementation of sensor network based autonomous vehicle control system," IEMEK J. Embed. Sys. Appl. Vol. 7, No. 5, pp. 247-253, 2012 (in Korean).

[3] S.H. Kim, "3D distance measurement of stereo images using web cams," IEMEK J. Embed. Sys. Appl. Vol. 3, No. 3, pp. 151-157 2008 (in Korean).

[4] C.H. Lee, "Investigation on the real-time environment recognition system based on stereo vision for moving object," IEMEK J. Embed. Sys. Appl. Vol. 3, No. 3, pp. 143-150 2008 (in Korean).

[5] G.Y. Jeong, "Efficient implementation of candidate region extractor for pedestrian detection system with stereo camera based on GP-GPU," IEMEK J. Embed. Sys. Appl. Vol. 8, No. 2, pp. 121-128, 2013 (in Korean).

[6] P. Dollar, "Pedestrian detection: an evaluation of the state of the art," IEEE Trans. Pattern Anal. Mach. Intell., Vol. 32, No. 4, pp. 743-761. 2012.

[7] D. Geronimo, "Survey on pedestrian detection for advanced driver assistance systems," IEEE Trans. Pattern Analysis and Machine intelligence, Vol. 32, No. 7, pp. 1239-1258, 2010.

[8] N. Dalal, "Histograms of oriented gradients for human detection," Proceedings of IEEE Conference on Computer Vision and Pattern

Recognition, 2005.
[9] M. Enzweiler, “Monocular pedestrian detection: Survey and experiments,” IEEE Trans. Pattern Anal. Mach. Intell, Vol. 32, No. 12, pp. 2179-2195, 2009.
[10] K, Grag, "Detection and removal of rain from videos," Proceedings of IEEE CVPR,, Vol. 1, pp. 528-535 2004.
[11] P. Barnum, "Analysis of rain and snow in frequency space," International Journal of Computer Vision, Vol. 86, No. 2-3, pp. 256-274, 2009.
[12] L.W. Kang, "Automatic single-image-based rain streaks removal via image decomposition," IEEE Transcaction on Image Processing, Vol. 21, No. 4, pp. 1742-1755, 2012.
[13] C. Tomasi, “Bilateral filtering for gray and color images,” Proceedings of IEEE International Conference on Computer Vision, pp. 839-846, 1998.
[14] J. Mairal, F. Bach, J. Ponce, G. Sapiro, “Online learning for matrix factorization and sparse coding,” The Journal of Machine Learning Research, Vol. 11, pp. 19-60, 2010.
[15] K. Alsabti, “An efficient k-means clustering algorithm,” Syracuse University, 1997.
[16] J. Sanders, *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Addison-Wesley, 2010.
[17] J. Reinders, *Intel Threading Building Blocks: Outfitting C++ for Multi-Core Processor Parallelism*, O'REILLY, 2007.
[18] http://darkpgmr.tistory.com/53

저 자 소 개

박 민 웅

2013년 인천대학교 임베디드시스템공학과 학사.
현재, 인천대학교 임베디드공학과 석사과정.
관심분야: 반도체 검사 영상처리 알고리즘, 병렬처리 시스템
Email: minwoong@incheon.ac.kr

정 근 용

2012년 인천대학교 임베디드시스템공학과 학사.
2014년 인천대학교 임베디드시스템공학과 석사.
현재, 옵토레인 부설연구소 연구원.
관심분야: 영상처리 알고리즘, 병렬처리 시스템
Email: keunyong.jung@optolane.com

조 중 휘

1981년 한양대학교 전자공학과 학사.
1983년 한양대학교 전자공학과 석사.
1986년 한양대학교 전자공학과 박사.
현재, 인천대학교 임베디드시스템공학과 교수.
관심분야: SoC 구조 및 설계, 영상신호처리 및 컴퓨터 비젼, 병렬처리 시스템
Email: jcho@incheon.ac.kr