JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 20, No. 10, pp. 2087-2096, Oct. 2019

전 · 후처리를 이용한 딥러닝 기반의 주차여부인식

김 대 진[1] · 윤 창 표[2] · 황 치 곤[3*]
[1]동국대학교 영상문화콘텐츠연구원
[2]경기과학기술대학교 컴퓨터모바일융합과
[3]광운대학교 정보과학교육원 컴퓨터공학과

Deep Learning based Parking Occupation Detection using Pre/Post-processing

Dae-Jin Kim[1] · Chang-Pyo Yoon[2] · Chi-Gon Hwang[3*]

[1]Research Institute for Image & Cultural Contents, Dongguk University, Seoul 04626, Korea

[2]Department Of Computer & Mobile Convergence, GyeongGi University of Science and Technology, Siheung-si 15073, Korea

[3]Department of Computer Engineering, Institute of Information Technology, Kwangwoon University, Seoul 01897, Korea

[요 약]

최근 주차공간의 효율적 관리를 위해서 주차유도 시스템이 점점 보급화 되고 있다. 단순히 주차할 곳을 찾기 위한 안내용으로 사용되기도 하지만, 차량 운전자가 본인의 차량이 주차된 곳을 찾기 위해서 영상처리 기술을 이용하여, 주차된 차량 찾기 서비스까지 연동되기도 한다. 따라서, 다양한 영상처리 및 패턴 인식 기술을 이용하여 주차여부인식 및 차량 번호 인식에 대한 연구가 지속되고 있다. 본 논문에서는 인식률을 높이면서, 빠르게 주차면의 주차여부인식을 할 수 있는 알고리즘을 제안한다. 주차면의 주차여부를 분석하기 위해서 전처리 부분으로 다중 임계치 병렬적용을 하였고, 보우팅 방법을 통해 객체 인식률을 높였으며, 딥러닝 기술(YOLO)을 이용한 카메라내 객체를 추출을 통하여 사람과 같은 다른 객체 추출에 의해서 발생할 수 있는 주차여부의 오류율을 줄일 수 있었다. 또한 인식률을 저하 시킬 수 있는 요인(빛, 장소)등에서도 제안한 알고리즘을 통한 높은 인식률을 얻을 수 있었다.

[Abstract]

Recently, parking guidance systems have been increasingly popular for efficient management of parking spaces. It is often used as an insider's guide to find a place to park, but it can also be linked to a parked vehicle search service using a camera-like image processing technology to find where the driver has parked his vehicle. Therefore, researches on parking occupation recognition and licence plate recognition using various image processing and pattern recognition technologies are continuing. In this paper, we propose an algorithm that recognizes parking occupation detection quickly in addition to increase the recognition rate. In order to analyze whether the parking slot is occupied, multiple thresholds are applied in parallel as a pre-processing part. Object recognition rate is increased through the voting method. Extraction of objects in the camera use deep learning(YOLO). It was possible to reduce the error rate of possible parking. Also, we can obtain high recognition rate through the proposed algorithm even in the factors that may decrease the recognition rate (light, circulation).

색인어 : 주차여부인식, 다중 임계치, 보우팅, 전처리, 딥러닝

Key word : Parking Occupation Detection, Multiple Thresholds Filtering, Voting, Pre-Processing, Deep Learning

http://dx.doi.org/10.9728/dcs.2019.20.10.2087

Received 26 August 2019; **Revised** 16 September 2019
Accepted 20 October 2019

***Corresponding Author; Chi-Gon Hwang**

Tel: +82-2-940-8379
E-mail: duck1052@kw.ac.kr

Ⅰ. 서 론

차량의 사용이 일반화됨에 따라 차량은 증가하고, 효율적인 주차를 위한 서비스들이 생겨나고 있다. 가장 일반화된 서비스는 자동차번호인식(LPR : License Plate Recognition)기술로 주차문화를 바꾸는 계기가 되고 있다[1]. 또한 최근에는 주차공간의 효율적 관리를 위해서 주차유도시스템이 점점 보급화 되고 있다. 주차유도시스템은 주차위치를 기반으로 차량 운전자가 빈 주차면에 빠르고 편안하게 주차할 수 있도록 유도를 하고, 주차 위치를 안내해 줌으로써 고객 서비스를 극대화하고, 무인 정산기 유도를 통해서 출구에서 정산하는 차량을 줄임으로써 주차장을 효율적으로 사용할 수 있게 도움을 주는 시스템이다[2]. 예전에는 초음파를 기반으로 한 시스템이었으나, 요즘은 그림 1과같이 주차장내에서 내 차가 주차된 위치까지 서비스를 받을 수 있는 영상 기반의 주차유도시스템이 대중화 되고 있다.

주차유도시스템에서 가장 기본 기술은 주차여부인식(Parking Occupation Detection)이다. 주차여부인식에 대한 연구들이 많이 진행 되고 있는데, R. Yusnita등은 이진화(Binary) 필터, 모폴로지(Morphology)등을 통해 형태분석을 하고 이를 통해 주차여부를 판단하였으며[3], Soomok Lee 등은 line marking을 이용하여 주차여부를 판단하였다[4]. Debaditya Acharya는 CNN(Convolution Neural networks)과 SVM(Support Vector Machine)을 이용하여 외부환경에서 날씨와 상관없이 주차여부를 판단하는 연구를 진행하였다[5]. 이를 기반으로 자동차 번호인식과 연동하여 단순히 주차할 곳을 찾기 위한 안내용으로 사용뿐만 아니라, 주차된 차량 찾기 서비스까지 연동되기도 한다.

본 논문에서는 주차유도시스템에서 주차여부인식률을 향상시키기 위한 여러 가지 알고리즘에 대하여 연구하였다. 기존의 알고리즘은 주차면의 객체 추출을 통해서 주차여부를 결정하였으나, 환경/시설 등에 의해서 인식률 낮아질 수 있기 때문에 여러 조합을 이용하여 주차여부를 판단하도록 해야 한다. 본 논문의 구성은 2장에서 기존 시스템에 적용된 객체 추출에 의한 주차여부인식을 분석하고 문제점에 대해 알아본다. 3장에서는 스탑퍼(Stopper) 객체 추출, 다중 임계치 병렬 적용, 보우팅(Voting)을 이용한 주차면 인식 및 YOLO를 통한 객체인식을 통해 주차여부인식률을 높이는 방법을 연구하였다. 4장에서는 제안한 알고리즘을 기반으로 하여 구현해 보고 5장에서 결론을 지었다.

Ⅱ. 기존 시스템 구현

2-1 객체 추출에 의한 주차면 주차여부 분석

그림 2에서는 주차유도시스템에서 사용하는 일반적인 객체 추출 기법을 이용한 주차면 주차여부 분석 흐름도를 나타낸다.

각 블록의 성능에 따라 속도 및 정확도가 좌우 된다. 흐름도 내 각 블록에 대한 설명은 아래와 같다.

1) 카메라 입력

주차유도 시스템 내에서 한 카메라에서 여러 개의 주차면을 바라보고 있다. 일반적으로는 주차면의 앞쪽에서 카메라가 3~4개의 주차면을 바라보고 있지만, 요즘은 비용적인 부분을 줄이기 위해서 주차면을 양쪽에 두고, 통로에 어안 렌즈 카메라를 배치하고, 양쪽 주차면을 모두 바라보게 한 후, 영상처리를 통해서 영상을 개선하여 사용하기도 한다. 그림 3에서는 3개의 주차면을 바라보고 있는 한 카메라 입력 이미지를 나타낸다.

2) 주차영역 분할

그림 3과 같이 한 카메라로부터 3개의 주차면을 바라보면, ①②③ 에 해당하는 각각의 주차면으로 분할하고, 주차영역을 분석하여 영역별 주차여부를 분석한다.

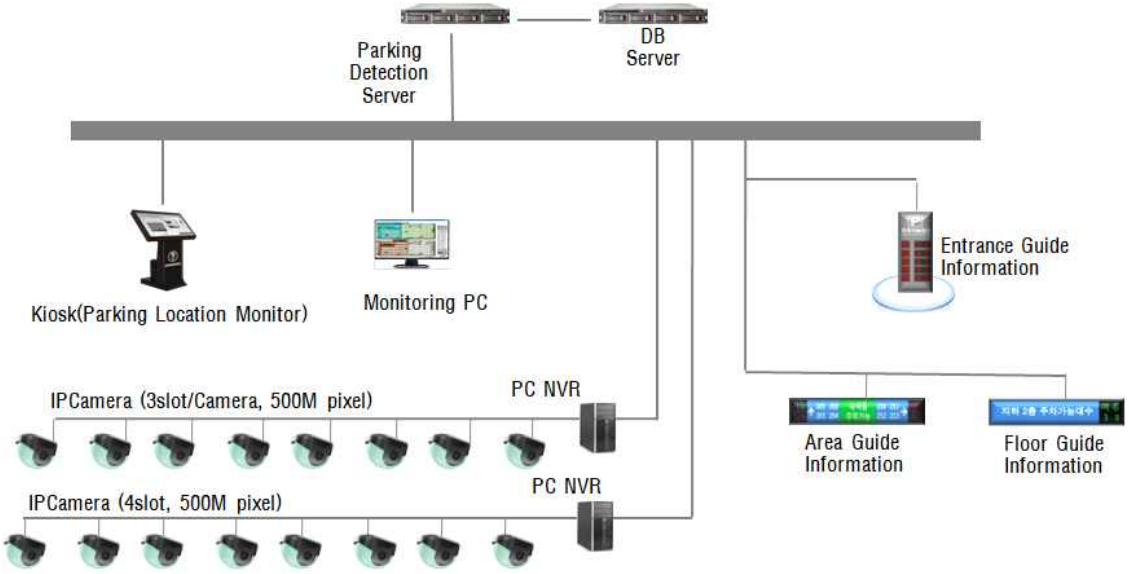

그림 1. 영상기반의 주차유도시스템 구성도
Fig. 1. Parking Guide System Architecture based Camera

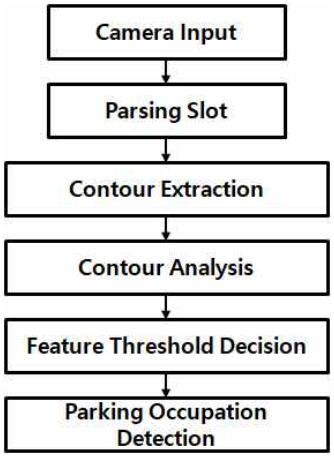

그림 2. 주차여부 분석 흐름도
Fig. 2. Vehicle Occupation Analysis Flow Chart

그림 3. 카메라입력을 통한 3개의 주차면
Fig. 3. three pieces of Parking Slot from Camera Input

3) 영역별 Contour 추출

분할 된 주차영역 별로 각각의 Contour를 추출한다. 추출된 Contour를 통해서 해당 주차면에 차량이 있는지를 분석 할 수 있는 데이터로 사용한다. 그림 4에서는 한 카메라로부터 입력 받은 3개의 주차면에 대한 Contour 추출 이미지를 나타낸다.

4) 영역별 Contour 분석

추출된 Contour를 통해서 주차영역안에 차량이 주차되어 있는지 분석을 해야 된다. 빈 주차면의 경우 주차면으로부터 추출할 수 있는 Contour의 개수가 적고, 빈 영역을 바라보고 있기 때문에 크기 또한 적고 한정적이다. 따라서 이 영역을 분석하면, 차량이 주차되어 있는지를 분석 할 수 있다.

5) 특징 임계치 판정

각 주차영역에 주차 여부를 판단하기 위해서 2가지 기준치를 사용하였다.

(1) Contour 개수

그림 4. 3개의 주차면에 대한 Contour 추출
Fig. 4. Contour Extraction in three pieces of Parking Slot

주차 가능한 상태인 빈 영역에서 추출된 Contour의 경우 그 개수가 한정적이다. 주차장 마다 조명, 시설물 환경이 다르기 때문에 차량이 있을 때와 없을 때를 구분하기 위한 Contour의 개수는 가변화 될 수밖에 없다. 따라서 해당 주차장 설치 시 이 기준치를 설정해 주어야 한다. 테스트한 주차장의 경우 Contour의 크기와 상관없이 50개 미만의 경우 빈 주차 공간으로 판단하였다.

(2) Contour 크기

Contour의 개수만으로 주차 여부를 판단하기에는 데이터가 불명확 할 수 있다. 비가 오는 날에는 주차면에 이물질이 많이 묻어 있을 수 있고, 조명 상태에 따라 Contour의 개수는 가변적이다. 따라서 Contour의 크기도 주차여부를 판단하기 위한 중요한 특징으로 사용해야만 한다. 테스트한 주차장의 경우 입력받은 영상은 4096x2160 pixel이며 이중 한 주차면에서 얻을 수 있는 Contour의 크기가 가로 400pixel, 세로 120pixel 이상이고 이 조건의 Contour가 3개 이상이 될 때 차량이 주차되어 있다고 판단하였다.

6) 영역별 주차여부 판단

임계치 판정에 따라 주차여부가 판단되면 주차장에 들어오는 고객들에게 이 사실을 알려야 한다. 현재 만공등을 통해서 해당 주차영역에 주차가능한지를 알린다. 초록색 등인 경우 주차가능하고, 빨간색 등인 경우 만차가 되었음을 알린다.

2-2 기존 시스템의 문제점 제시

일반적인 상황인 경우 기존의 알고리즘을 사용해도 문제없이 사용 할 수 있다. 그러나, 서비스 성능을 높이기 위해서는 예외적인 상황이 발생 했을 때도 성능을 높일 수 있는 방안이 필요하다. 문제가 발생할 수 있는 조건은 아래와 같다.

1) 빛에 의한 간섭

영상처리 알고리즘에서 가장 중요한 요소는 입력 데이터의 분석 가능한 형태로의 습득이다. 이중 과도한 빛, 반사 또는 고르지 않은 빛의 상태 등은 입력영상을 손상시키는 계기가 된다. 그림 5에서는 주차면의 빛의 간섭을 보여준다. 주차시설의 구조 또는 다른 차량에 의해서 주차면마다 과도한 빛의 반사가 이

그림 5. 주차면의 빛의 간섭
Fig. 5. Light Interference in Parking Slot

루어져 각 주차면마다 다른 주차면 상태를 나타낸다.

2) 사람, 다른 차량 등 다른 객체의 간섭

카메라로 차량을 인식해야 되는 경우 다른 객체의 의해서 입력영상의 간섭이 일어나는 경우가 많이 발생한다. 차량들은 주차할 곳을 찾아 계속 이동하기 때문에 이동하는 순간에 카메라를 방해하게 된다. 그림 6에서는 주차여부인식률에 영향을 줄 수 있는 다른 객체의 간섭 경우를 나타낸다. 다른 차량이 통로를 이동하거나, 다른 차가 주차할 때 주차면을 가리거나, 사람이 이동시 차량을 가리거나, 공사 차량이 2개의 공간을 차지하고 있는 경우 등 다양한 경우에 간섭이 발생할 수 있다.

그림 6. 주차면에서 다른 객체의 간섭
Fig. 6. Other Object Interference in Parking Slot

3) 특수 주차면에 따른 간섭

빈 주차면이라도 주차 면 마다 항상 다른 상태를 유지하게 된다. 장애인 주차영역 또는 에너지 1등급 차량 등의 주차면에는 특정 이미지로 표시되어 안내하고 있다. 이 경우 주차여부를 판단할 때 Contour를 이용하면 주차 가능한 상태일지라도 주차 불가로 나올 수 있다. 그림 7에서는 특수 주차면의 경우를 나타낸다.

4) 그 밖의 간섭

1) 2) 3) 이외에도 비오는 날의 빗물 얼룩, 시설물 구조에 의한 영향, 사람이 주차면에 계속 있는 경우, 공사에 의해서 시설물을 주차공간에 임시로 넣는 경우, 뒤 트렁크를 열어 두어 차

그림 7. 특수 주차면에서의 간섭
Fig. 7. Interference in Special Parking Slot

량이 차량으로 인식 되지 않는 경우 등 인식률에 영향을 줄 수 있는 수많은 경우의 수가 발생할 수 있다.

이와 같은 간섭 요소들에도 인식속도 및 인식률을 높일 수 있는 연구가 필요하다. 따라서 본 연구에서는 전·후처리를 이용한 딥러닝 기반의 주차여부인식에 대하여 제안한다. 일반적인 상황에서는 주차장에 있는 스탑퍼(Stopper)를 이용하여 주차면의 주차여부를 빠르게 분석 하였고, 빛에 대한 영향을 최소화하기 위해 전처리 부분에 다중 임계치를 병렬 적용 하였으며, 보우팅 방법을 통해 주차면 인식 정확도를 높일 수 있었다. 또한 딥러닝 기술인 YOLO를 통한 객체 인식을 통해 실제 상황에서 발생할 수 있는 오류율을 최소화 하는 연구를 진행하였다.

Ⅲ. 제안하는 전 · 후처리를 이용한 딥러닝기반의 주차여부인식

한 알고리즘 적용을 통하여 주차 이용 공간을 100% 인식하면 좋으나, 실제 환경에서는 거의 불가능하다. 따라서 각 상황에 맞는 적절한 알고리즘을 선별적 적용하여 인식률을 끌어 올리는 것이 적합하다. 따라서 아래와 같이 여러 가지 알고리즘을 조합하여 주차여부 인식을 할 수 있다.

3-1 스탑퍼를 이용한 주차면 분석

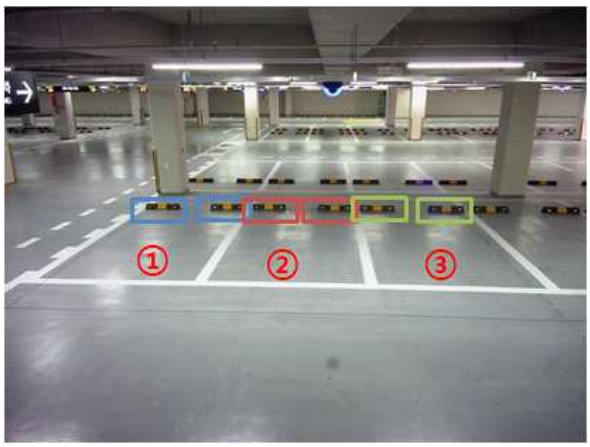

그림 8. 스탑퍼를 이용한 주차 영역 설정
Fig. 8. Parking Slot Setting using Stopper

요즘 주차장은 주차시 차량의 충돌을 방지하기 위하여 주차

영역내에 스탑퍼를 설치하고 있다. 이는 주차유무를 판달 할 수 있는 중요한 특징으로 사용할 수 있다. 카메라로부터 Contour를 추출하는 경우 영상 전체에서 특징을 뽑기 때문에 시스템 구성시 속도에 영향을 미치게 된다. 인식해야 될 주차면 그룹을

10초안에 모두 인식해야 한다는 가정아래, 한 대의 서버에서 60대의 카메라 영상을 인식하고, 한 카메라당 3대의 주차면을 바라본다면, 한 서버에 10초당 180개의 주차면을 인식하게 된다. 주차위치 안내를 위해서 보통 주차면 인식과 함께 차량번호 인식이 같이 적용되는데 인식에 200ms 정도 소요된다면 한 서버당 약90~100대 정도의 주차면을 인식에 사용 할 수 있다.

시스템 구성시 비용 감소는 중요한 요소이다. 따라서, 인식의 속도를 높이면서 시스템 구성 비용을 줄이기 위해서 전체 이미지가 아닌 스탑퍼를 이용한 인식을 진행하였다.

카메라를 통해서 인식영역을 설정할 때, 그림 8과 같이 주차면 전체가 아닌 스탑퍼 부분만 영역으로 설정하였다.

스탑퍼만 인식에 사용하는 경우 Contour 추출을 할 때 4096x2160 pixel의 이미지에서 추출할 때보다 약 1/10 속도가 줄기 때문에 10초당 1800개의 주차면을 인식하게 되고, 차량번호인식까지 적용하는 경우 한 서버당 약 500개 정도의 주차면을 인식하는데 사용할 수 있다.

알고리즘 적용시 기둥, 빛, 사람, 차량, 물건 등에 의해서 그림 9와 같이 한쪽 스탑퍼만 보이는 경우도 발생한다.

그림 10에서는 스탑퍼를 이용한 주차여부 판단 순서도를 나타낸다. 먼저 카메라로부터 주차면의 이미지를 얻게 된다. 이때 그림 8과 같이 설정한 주차면의 스탑퍼로부터 Contour를 추출

그림 9. 시설물에 의해 한쪽 스탑퍼만 보이는 경우
Fig. 9. Visible one-Stopper in Parking Slot by Facility

하게 된다. Contour 추출을 위해서 Canny Edge 알고리즘을 통해 Edge 부분을 먼저 찾은 후, 추출된 Edge를 기반으로 스탑퍼의 Contour를 추출한다[6][7][8]. Contour 추출을 위해 아래와 같은 과정을 가진다.

1) 노이즈 제거

5x5 가우시안 필터(Gaussian Filter)를 적용하여 이미지 필터링을 적용한다. 실내에 있는 카메라의 경우 시간이 지나면서 차량 매연 등에 의한 먼지가 쌓이거나, 주차장 청소시 얼룩등 다양한 경우로 카메라 또는 주차면 영상에도 노이즈로 발생할 수 있기 때문이다.

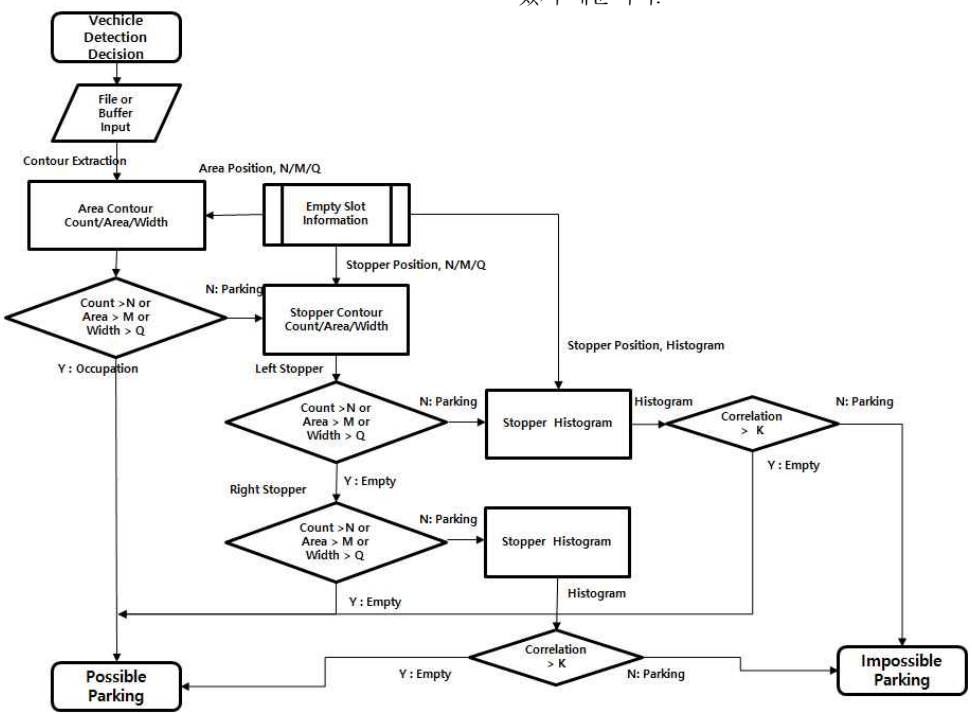

그림 10. 스탑퍼를 이용한 주차여부 판단 순서도
Fig. 10. Flow Chart of Parking Occupation Detection using Stopper

2) Gradient 값이 높은 부분 찾기

Sobel 커널(Kernel)을 수평방향, 수직방향으로 적용하여 각 방향의 Gradient를 획득한다. 수평 방향의 Gradient를 Gx, 수직 방향의 Gradient를 Gy로 할 때 픽셀 (x,y)에서 Edge Gradient는 아래와 같다.

$$Edge\,Gradient(G) = \sqrt{G_x^2 + G_y^2} \quad \text{..... (1)}$$

$$Angle(\theta) = \tan^{-1}\left(\frac{G_x}{G_y}\right) \quad \text{..... (2)}$$

3) 최대값이 아닌 픽셀의 값을 0으로 만들기

에지에 기여하지 않은 픽셀들은 이미지 스캔(Scan) 된다. 이때 Gradient 방향으로 스캔구역에서 최대값을 가진 픽셀을 찾는다.

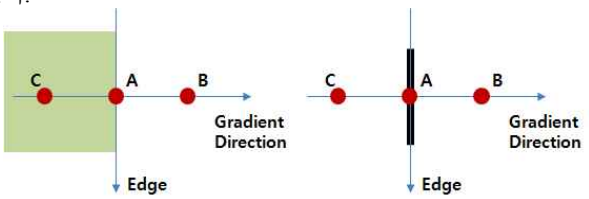

그림 11. Gradient를 통한 에지 체크
Fig. 11. Edge Check by Gradient

그림 11을 보면 A는 수직 방향의 에지위에 있는 픽셀이고, Gradient 방향은 수평방향이다. 이때 B와 C는 Gradient 방향에 놓인 픽셀로 A지점에서 Gradient 값이 B, C 보다 큰 값을 가지는지 확인 한다. A에서 값이 가장 크면 다음 단계로 넘어가고, 그렇지 않다면 0으로 만든다.

4) Hyteresis Thresholding

3) 을 통해 얻은 에지를 추적하여 실제 에지 여부를 판단하게 된다. 임계치(min, max)를 2개 설정하여 max값보다 큰 부분이나, min, max 사이의 있더라도 max와 연결되는 부분은 에지로 판단한다.

추출된 에지정보를 통해서 Contour를 추출 할 수 있다. 에지의 외각 점들을 연결하는 모든 Contour라인들을 찾고 이에 따른 상하구조(Hierarchy)관계를 구성한다. 이들 Contour들을 라인 형태로 연결하여 스탑퍼의 존재여부를 판단하는 특징으로 사용한다. 그림 12는 추출된 스탑퍼의 Contour를 보여준다.

미리 설정된 주차면의 스탑퍼로부터 Contour 정보를 추출한다. 해당 영역의 Contour의 개수/면적/가로길이 정보를 통해서 형태적인 정보로 스탑퍼의 유무를 판단한다. 그림 10의 순서도에서 스탑퍼 Contour 관련 정보는 아래와 같다.

N : 주차면에서 가져온 한 스탑퍼의 Contour 개수
M : 주차면에서 가져온 한 스탑퍼의 Contour 면적의 합
Q : 주차면에서 가져온 한 스탑퍼의 Contour 가로길이의 합
K : 한 스탑퍼의 히스토그램 연관성(Histogram Correlation) 기준치

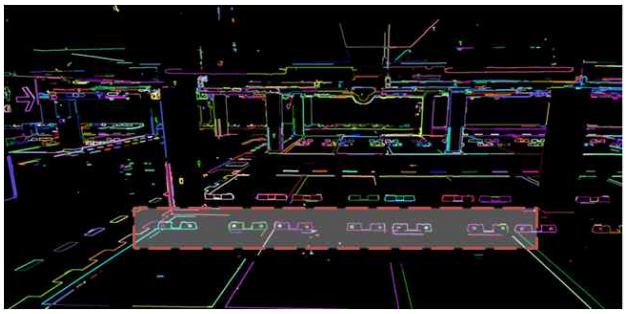

그림 12. 스탑퍼 Contour 추출
Fig. 12. Contour Extraction in Contour

또한 형태적인 정보로 스탑퍼를 판단할 때 다른 물건 등이 스탑퍼를 가리고 있는 경우 형태적인 면에서 스탑퍼처럼 판단할 수 있다. 따라서 색상정보를 통해서 스탑퍼 유무의 정확성을 높일 수 있다. 이를 위해 본 논문에서는 히스토그램 연관성(Histogram Correlation)을 사용하였다[9]. 이때 적용 방법은 아래와 같다.

$$d(H_1,H_2) = \frac{\Sigma_I(H_1(I)-\overline{H_1})(H_2(I)-\overline{H_2})}{\sqrt{\Sigma_1(H_1(I)-\overline{H_1})^2\Sigma_1(H_2(I)-\overline{H_2})^2}} \quad \text{...(3)}$$

$$\overline{H_k} = \frac{1}{B}\sum_J H_k(J) \quad \text{.....(4)}$$

B : 히스토그램 빈(Histogram Bin)의 전체 수

본 실험에서는 N=3, M=400, Q=200, K=0.7 으로 하였다. 이때 기준이 되는 스탑퍼의 히스토그램을 미리 가지고 있어야한다.

3-2 다중 임계치 병렬 적용을 이용한 영상 전처리

주어진 환경이 항상 일정하면 인식의 정확도를 높일 수 있다. 그러나, 먼지에 의한 카메라 이물질, 자동차 라이트 불빛, 구조물에 의한 그림자 등 다양한 간섭 요인들이 발생할 수 있다. 따라서 인식률을 높이기 위해 일정한 환경을 구성 하는 것이 중요하다. 이를 개선하기 위한 2가지 방법이 있다.

1) 카메라 설정 변경을 통한 영상 전처리

설치된 카메라에 성능에 따라 좌우되는 요소이다. 카메라의 경우 설정이 가능한 경우 이를 통해 전처리를 통해서 인식률을 높일 수 있다.

카메라에서 설정 가능한 부분은 노출(Exposure), 심도와 조리개, 셔터속도, 게인, 역광현상, HCC(Highlight Compensation Capability), WDR(Wide Dynamic Range), 노이즈(Noise), 화이트 밸런스(White Balance) 등이 있다. 이들에 대한 자동조정 기

(a)수동노출(Non-Auto Exposure) (b)자동노출(Auto Exposure)
그림 13. 카메라 노출 조정비교
Fig. 13. Camera Exposure Control Compare

능이 있다면, 최대한 이용하여 인식률을 높일 수 있다. 본 논문에서 테스트한 카메라의 경우 자동노출(Auto Exposure) 기능을 사용 하였다.

그림 13에서는 카메라 노출 조정여부에 따른 입력된 영상을 비교 하였다. 햇빛이 강하게 한곳을 비치는 경우 해당 부분에서 Contour를 추출 하지 못하는 경우가 발생할 수 있다. 따라서, 항상 빛의 노출이 과하지 않게 조정하는 것이 중요하다.

2) 다중 임계치 병렬 적용을 통한 영상 전처리

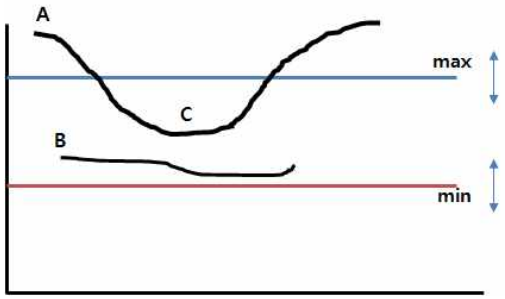

그림 14. 임계치 가변 적용
Fig. 14. Variable Threshold Application

카메라의 설정에 따라 HW 전처리를 했다면, 다중 임계치 병렬 적용을 통한 SW 전처리 과정으로 인식률을 높일 수 있다. Edge 검출시 Hyteresis Thresholding 처리를 위한 임계치(min, max)를 가변적으로 처리 한다면, 인식에 보다 적합한 영상을 얻을 수 있다.

임계치(min, max) = {(50, 150), (75, 175), (100, 200), (125, 225), (150, 250)}을 5개를 두고 이 기준에 맞추어서 Edge를 검출 하였다. 그림 14에서는 Edge 추출 시 임계치 가변 적용을 나타낸다.

최적의 입력상태를 유지하기 위해서 각각의 임계치를 적용했을 때 적합한 이미지를 선택해야 된다. 본 논문에서는 전처리된 이미지중 Contour의 N, M, Q가 반드시 한 개 이상 기준을 만족하면서 N, M, Q가 기준을 초과하는 경우의 개수의 합이 가장 큰 이미지를 선택하여 인식에 사용한다. 본 논문에서는 그림 8에서 주차영역의 경우 임계치(min, max)=(150, 250) 일 때 주차여부인식에 사용하였으며 그림 15에서는 인식에 적용한 스탑퍼의 Contour를 나타낸다. 선택된 스탑퍼는 그림 10의 과정을

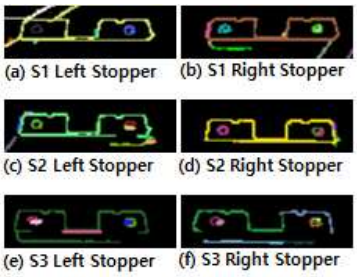

그림 15. 인식에 적용한 스탑퍼의 Contour
Fig. 15. Selected Stopper Contour for Recognition

통해서 주차여부를 판단한다.

3-3 보우팅(Voting)을 이용한 주차면 인식

그림 16. 실내 주차장에서 주차 유도 카메라 설치
Fig. 16. Camera install in Parking Guide System at inner Parking Place

만공등 맞은편에 있는 카메라가 바라보고 있는 파킹 슬롯의 3~4면을 바라보고 있다. 그림 16은 실내 주차장에서의 주차 유도 카메라 설치를 나타낸다.

카메라로 입력받은 영상은 그림 3과 같은 이미지를 얻게 되고 이를 통해 영역별 주차여부 판단을 진행한다. 이 결과에 따라서 그림 17과 같이 맞은편 주차면의 주차가능여부를 판단하고, 만공등을 통해서 주차 가능여부를 운전자에게 안내한다.

이상적인 상황에서는 주차 가능 여부를 판단하는데 문제가 발생하지 않으나, 현장에서 적용 시는 차의 이동, 사람의 이동 등 그림 6과 같은 다양한 상황들이 발생하게 된다. 이러한 상황이 발생 되더라도 주차여부 판단에 대한 정확한 판단이 필수적이다. 따라서, 본 논문에서는 주차면에 대해서 보우팅을 진행한다. 즉 주차공간에 시간적인 점유여부를 판단하여 최종 주차가능여부를 판단한다.

한 서버에서 최종주차여부(D)를 결정하기 위해 주기(P)를 10초로 정의 하였다. 이 기준치는 주차여부 최종 결정까지 서비스 이용에 문제가 없을 정도라 판단한 심리적 시간으로 실험적으로 결정하였다. 한 서버에 2초당 54개의 주차면을 인식하였고, 최종 주차여부(D)는 주기(P)에 대한 보우팅 결과로 결정하였다. 그림 18에는 보우팅을 이용한 주차면 인식에 대해서 나타낸다. 주차면 ①의 경우 주기(P)동안 모두 parking 상태이며 최종 주차 된 것으로 판단하였고, ②의 경우 empty 상태에서 4번

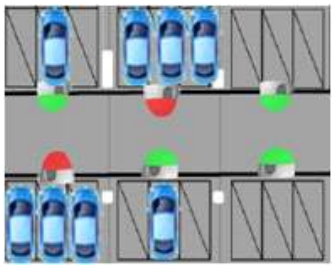

그림 17. 주차면과 만공등
Fig. 17. Parking Slot and Full/Empty Lamp

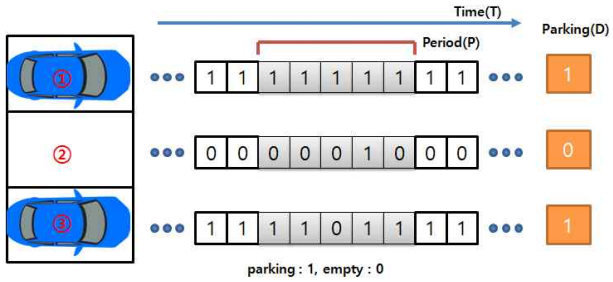

그림 18. 보우팅을 이용한 주차면 인식
Fig. 18. Parking Slot Recognition using Voting

째에 외부요인의 간섭에 의해 parking 상태로 잠시 바뀌었으나 (parking : empty) = (1:4)의 보우팅 결과에 의해서 최종 주차되지 않은 것으로 판단하였다. ③의 경우 주기(P)동안 parking 상태이지만 3번째 체크시 잠시 empty 상태로 잠시 변경이 있으나 (parking : empty) = (4:1)로 최종 주차된 것으로 판단하였다. 이와 같이, 주차장에서는 정형화되지 않은 간섭요인들이 항상 많이 발생되기 때문에 보우팅 알고리즘을 이용하는 것은 필수적인 요소이며, 그 결과 주차여부 인식률을 높일 수 있었다.

3-4 YOLO를 이용한 객체(자동차, 사람) 인식

관심 영역 기반의 검출 기법들이 많이 연구되고 있다. R-CNN[10]은 Selective Search 알고리즘을 기반으로 입력 영상 내 객체 후보 영역(Region Proposal)을 제안 후 각 영역에 대하여 CNN(Convolution Neural Network)을 거쳐 SVM(Support Vector Machine) 분류기를 통한 영역 내 객체 패턴의 클래스를 분류한다. Faster R-CNN[11]은 RoI(Region of Interest) Pooling의 개념을 도입하여 Selective Search에서 찾은 바운딩 박스(Bounding Box)정보가 CNN을 통과하면서 유지되고 최종 CNN 특성 맵(Map)으로 부터 해당 영역을 추출하여 풀링(Pooling)한다. 그러나, ROI마다 객체 분류를 수행해야 하는 동작 원리 때문에 전체 검출기의 수행 속도가 저하되는 요인이 된다. 따라서, 객체 인식률을 유지하면서 동시에 검출 속도를 향상 시키려는 다양한 연구들이 시도되었다. 그 중 최근 가장 주목을 받은 방법인 YOLO(You Only Look Once)알고리즘을 이용한 객체 인식으로, 네트워크는 최종 출력 단에서 경계 상자검출과 클래스 분류가 동시에 수행되도록 설계되었다[12][13].

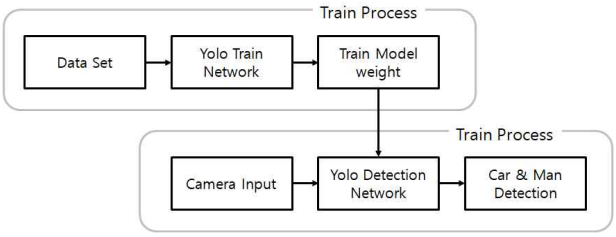

그림 19. YOLO를 이용한 객체 학습 및 검출 과정
Fig. 19. Object Train & Detection Processing using YOLO

본 논문에서 YOLO를 이용한 객체 학습 및 검출과정은 그림 19와 같다. 주차장내에 검출해야 될 객체를 자동차와 사람으로 2가지로 Class를 구성하였으며, YOLO의 Custom Object학습 방법을 통해서 2가지(자동차, 사람) 객체로 학습하였다[14].

이때 구성한 YOLO 네트워크 구성은 표 1과 같다. 19개의 Convolution layer와 5개의 Max Pooling Layer로 네트워크를 구성하였으며, Fully Connected Layer를 Global Average Pooling을 사용하여 Parameter 수가 증가되는 것을 방지하고 연산속도를 증가시켰다. 그림 20에서는 주차장내 YOLO를 이용한 객체 검출 결과물을 보여준다.

표 1. YOLO 네트워크 구성[15]
Table. 1. YOLO Network Model

Type	Filters	Size/Stride	Output
Convolution	32	3 x 3	224x 224
Maxpool		2 x 2/2	112 x 112
Convolution	64	3 x 3	112 x 112
Maxpool		2 x 2/2	56 x 56
Convolution	128	3 x 3	56 x 56
Convolution	64	1 x 1	56 x 56
Convolution	128	3 x 3	56 x 56
Maxpool		2 x 2/2	28 x 28
Convolution	256	3 x 3	28 x 28
Convolution	128	1 x 1	28 x 28
Convolution	256	3 x 3	28 x 28
Maxpool		2 x 2/2	14 x 14
Convolution	512	3 x 3	14 x 14
Convolution	256	1 x 1	14 x 14
Convolution	512	3 x 3	14 x 14
Convolution	256	1 x 1	14 x 14
Convolution	512	3 x 3	14 x 14
Maxpool		2 x 2/2	7 x 7
Convolution	1024	3 x 3	7 x 7
Convolution	512	1 x 1	7 x 7
Convolution	1024	3 x 3	7 x 7
Convolution	512	1 x 1	7 x 7
Convolution	1024	3 x 3	7 x 7
Convolution Average Softmax	1000	1 x 1 Global	7 x 7 1000

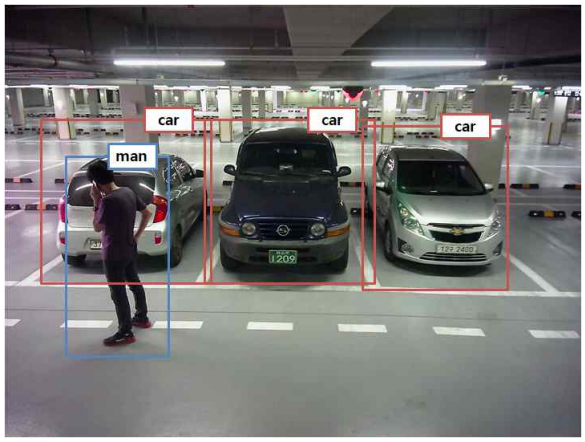

그림 20. 주차장내 YOLO를 이용한 객체 검출
Fig. 20. Object Detection using YOLO at Parking Lot

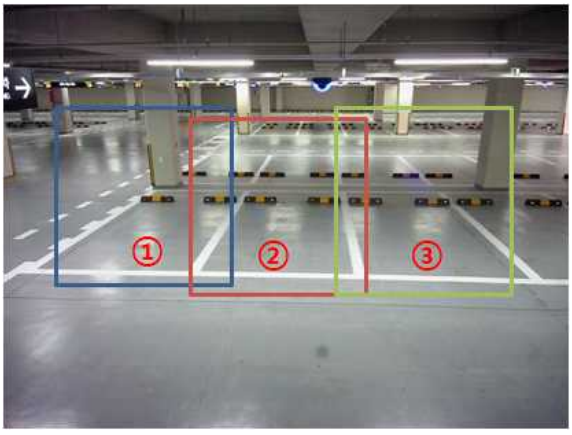

그림 21. YOLO 객체 검출을 위한 주차 영역 설정
Fig. 21. Parking Slot Setting for YOLO Detection

주차면에 주차가 된 것을 확인하기 위해서 그림 21처럼 주차 영역이 미리 설정 되어 있어야 한다. 영역을 설정할 때 실제 주차된 차량이 카메라 각도 상 옆의 주차면까지 점유할 수 있으므로 좌/우 주차면까지 겹치도록 충분한 크기의 영역을 설정해야 주차여부인식 결과가 보다 정확하게 나올 수 있다.

주차여부판단은 설정된 인식영역대비 YOLO에 의해서 차량으로 판정된 객체 인식영역의 75% 이상 차지하면 주차로 판정하였다. 여기서 75%는 실험치로 주차장 환경에 따라 초기 설정값을 결정하면 된다.

Ⅳ. 시스템 구현 결과

4-1 시스템 구현환경

기존의 주차여부인식 알고리즘과 다중 알고리즘을 적용했을 때 인식률을 비교하기 위해 7일 동안 98개의 카메라에서 300개의 주차면에서 촬영한 약 15만개의 이미지를 분석해 보았다. 카메라로부터는 4096x2016 pixel의 이미지를 습득하였으며, 1개의 카메라에서 2~4개의 주차면을 바라보고 있다. 테스트 기간 중 비가 내리지 않았기 때문에 비에 의한 오인식 영향 보다는 빛, 조명, 시설물에 의한 영향이 더 있었다. 인식엔진 장비는 OS가 64bit Windows10에 Intel(R) core(TM) i5-7500 CPU@3.40GHz, 메모리는 16Gbyte를 사용하였으며, YOLO를 이용한 객체 검출시 GPU를 사용하였고, Graphic Card는 RTX 1080TI를 사용하였다. CPU 인식서버는 4대를 사용하였으며, YOLO구현시 darknet[16]을 사용하여 구현하였으며, YOLO 엔진 사용 시 GPU 인식서버 1대 운영하였다. CPU 인식서버는 스탑퍼 객체 추출, 다중 임계치 병렬 적용, 보우팅(Voting)을 이용한 주차면 인식을 할 때 사용하였으며, 보우팅시 parking/empty 비율이 2:3 또는 3:2인 경우 정확한 인식 결과를 도출하기 위해 GPU 인식 서버를 사용하였다. 인식 서버 개발 시 WAS(Web Application Server)는 Visual Studio 2015 C++로 개발 하였다.

4-2 시스템 성능비교

표 2에서는 기존 알고리즘과 제안한 알고리즘의 속도 및 인식률을 비교하였다. 100개의 주차면에 대해서 실험했을 때 기존 알고리즘과 CPU기반 알고리즘과의 비교시 인식률이 약 10%, 속도 약7.8배 정도 개선되었고, 기존 알고리즘과 GPU기반 알고리즘 비교시 인식률이 약17%, 속도 약3.7배 정도 개선되었다. 이때 원 카메라 입력영상의 크기로 알고리즘을 사용하는 경우 메모리에러가 발생하여 1920x1080으로 크기변환 후 사용하였다. 기존알고리즘과 CPU+GPU기반 알고리즘과의 비교시 인식률이 약17%, 속도 약4.8배 정도 개선되었다. 전반적으로 GPU기반의 YOLO알고리즘을 사용했을 시에 인식률의 강점을 보였다. 그러나, GPU를 사용하였기 때문에 Graphic Card의 비용적인 부분에 약점을 보였다. 따라서, 실제 현장 적용시 이 두 부분을 해결할 수 있는 시스템 설계가 필요하다.

표 2. 알고리즘 비교
Table. 2. Algorithm Comparison

Algorithm	ARR (%)	ARS (sec)
Object Detection	81.41	16.67
CPU base	91.53	2.12
GPU base	98.76	4.53
CPU+GPU base	98.89	3.47

CPU base = Stopper Object Detection + Multiple Thresholds + Voting Method
GPU base = YOLO Algorithm
ARR(Average Recall Rate) = Average Recognition Rate in 300 Parking Slots during 7 Days
ARS(Average Recognition Speed) = Average Recognition Speed in 300 Parking Slots during 7 Days

Ⅴ. 결 론

본 논문에서는 실제 현장에서 적용할 수 있는 주차여부인식

알고리즘을 제안한다. 현장에서 적용할 때 가장 중요한 것은 인식률, 처리속도, 비용절감이다. 인식률만 고려한다면 YOLO를 이용해서 주차여부판단을 하는 것이 가장 효율적이나, GPU를 사용하기 위해 고비용의 그래픽카드를 사용해야 되는 이슈가 있기 때문에 이 부분을 고려한 시스템이 필수적이다. 따라서 제안하는 알고리즘을 통해 여러 제약 사항들에서 효율적으로 대처가능 하면서, 전체 시스템 비용을 줄이고 빠르게 인식하면서 인식률을 향상 시킬 수 있었다.

참고문헌

[1] Dae-Jin Kim, “Implementation of Parking Management System using Cloud based License Plate Recognition Service," *Journal of Digital contents Society*, Vol. 19, No. 1, pp. 173-179, 2018.

[2] L. Li, L. Zhang, X. Li, X. Liu, Y. Shen, and L. Xiong, “ Vision-based parking-slot detection: A benchmark and a learning-based approach,” *in Proc. IEEE Int. Conf. Multimedia Expo* , pp. 649–654, Jul. 2017.

[3] Yusnita, R., Norbaya, F. and Basharuddin, N., “Intelligent Parking Space Detection System Based on Image Processing.” *International Journal of Innovation, Management and Technology*, 3, pp. 232-235, 2012

[4] Lee, S.; Seo, S., “Available parking slot recognition based on slot context analysis”, *IET Intel. Transp. Syst*. 10, pp.594-604, 2016

[5] Debaditya Acharya, Weilin Yan, Kourosh Khoshelham, “Real-time image-based parking occupancy detection using deep learning”, *Proceedings of the 5th Annual Conference of Research@Locate*, Vol. 2087, pp. 33–40, 2018

[6] Canny Edge Detection[Internet]. Available: https://en.wikipedia.org/wiki/Canny_edge_detector

[7] Moeslund, T. (2009, March 23). Canny Edge Detection. Retrieved December 3, 2014

[8] Python Canny Edge Detection[Internet]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html

[9] Histogram Camparison[Internet]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580-587, 2014

[11] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 39, pp. 1137-1149, Jun, 2016.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *In Proc. CVPR*, 2016.

[13] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. arXiv, 2018.

[14] Darknet Custom Object Train[Internet]. Available : https://github.com/AlexeyAB/darknet#how-to-train-to-detect -your-custom-objects

[15] JJoseph Redmon, Ali Farhadi, “YOLO9000: Better, Faster, Stronger.”, University of Washington, Allen Institute for AI, CVPR 2017.

[16] J. Redmon. Darknet: Open source neural networks in c.[Internet]. Available : http://pjreddie.com/darknet/

김대진(Dae-Jin Kim)

1998년 : 대진대학교 대학원(공학사)
2000년 : 동국대학교 대학원 (공학석사)
2010년 : 대진대학교 대학원 (공학박사)

2017년～현 재: 동국대학교 영상문화콘텐츠연구원 조교수
※관심분야 : 코덱, 멀티미디어 플랫폼, 콘텐츠 DNA, 워터마크,딥러닝, 번호인식, 주차관제시스템 등

윤창표(Chang-Pyo Yoon)

1998년 : 광운대학교 전자계산학과 (이학사)
2001년 : 광운대학교 컴퓨터과학과 (공학석사)
2012년 : 광운대학교 컴퓨터과학과 (공학박사)

2012년～현 재: 경기과학기술대학교 컴퓨터모바일융합과 교수
※관심분야 : 기계학습, 모바일 시스템, 네트워크 보안, 무선 네트워크, 온톨로지

황치곤(Chi-Gon Hwang)

1995년:창원대학교 경영학과 (경영학사)
2004년:광운대학교 정보통신학과(공학석사)
2012년:광운대학교 컴퓨터과학과 (공학박사)

2006년～2015년:(주)인찬 연구원
2016년～2018년: 경민대학교 인터넷정보과 교수
2019년～현 재: 광운대학교 정보과학교육원 컴퓨터공학과 교수
※관심분야 : 모바일 클라우드, 멀티미디어 온톨로지, 클라우드 컴퓨팅, 데이터 상호운용